TRƯỜNG ĐẠI HỌC KINH TẾ QUỐC DÂN
VIỆN CÔNG NGHỆ THÔNG TIN VÀ KINH TẾ SỐ
TS. PHẠM XUÂN LÂM (Chủ biên)
ThS. PHẠM ĐỨC TRUNG
ThS. CAO THỊ THU HƯƠNG
ThS. CHU VĂN HUY
****

Giáo trình
THIẾT KẾ WEB

Nhà xuất bản Đại học Kinh tế Quốc dân

TRƯỜNG ĐẠI HỌC KINH TẾ QUỐC DÂN
VIỆN CÔNG NGHỆ THÔNG TIN VÀ KINH TẾ SỐ
TS. PHẠM XUÂN LÂM (Chủ biên)
ThS. PHẠM ĐỨC TRUNG
ThS. CAO THỊ THU HƯƠNG
ThS. CHU VĂN HUY
****

Giáo trình
THIẾT KẾ WEB

Nhà xuất bản Đại học Kinh tế Quốc dân

MỤC LỤC

MỤC LỤC **3**
DANH MỤC TỪ VIẾT TẮT **12**
DANH MỤC HÌNH **13**
DANH MỤC BẢNG **17**
LỜI NÓI ĐẦU **18**
CHƯƠNG 1: GIỚI THIỆU VỀ THIẾT KẾ WEBSITE **21**
1.1 TỔNG QUAN VỀ THIẾT KẾ WEBSITE **21**
1.1.1 Website là gì? 21
1.1.2 Mạng máy tính 21
1.1.3 Phân loại Web 22
1.1.4 Đường dẫn Web 24
1.1.5 Trình duyệt Web 25
1.2 QUY TRÌNH THIẾT KẾ WEBSITE **26**
1.3 CÁC VẤN ĐỀ CẦN QUAN TÂM KHI THIẾT KẾ WEBSITE **31**
1.3.1 Người dùng và tính khả dụng 32
1.3.2 Khả năng tương thích của trang Web 32
1.3.3 Khả năng tiếp cận mọi đối tượng người dùng 33
1.3.4 Tối ưu hóa công cụ tìm kiếm 33
1.3.5 Thiết kế Web đáp ứng 33
1.4 HTML, CSS, JAVASCRIPT **35**
1.4.1 Tài liệu HTML 35
1.4.2 Ngôn ngữ định dạng CSS 37
1.4.3 JavaScript 39
1.5 CÔNG CỤ THIẾT KẾ GIAO DIỆN WEB **40**
1.5.1 Bộ công cụ của Adobe 40
1.5.2 Công cụ Figma 41
1.6 CÔNG CỤ VIẾT MÃ LỆNH THIẾT KẾ **41**
1.6.1 Công cụ Visual Studio Code 41
1.6.2 Công cụ Sublime Text 42
1.6.3 Bộ công cụ của JetBrains 43
1.6.4 Công cụ CodePen.io 43
1.6.5 Công cụ CodeSandbox.io 44
1.6.6 Công cụ jsFiddle.net 45
1.7 CÔNG CỤ HỖ TRỢ PHÁT TRIỂN VÀ GỠ RỐI TRÊN TRÌNH DUYỆT **45**
CÂU HỎI ÔN TẬP LÝ THUYẾT **47**
BÀI TẬP TỰ THỰC HÀNH **48**
TÀI LIỆU THAM KHẢO **48**
CHƯƠNG 2: NGÔN NGỮ ĐÁNH DẤU SIÊU VĂN BẢN HTML **49**
2.1 TÀI LIỆU HTML **49**
2.1.1 Giới thiệu về ngôn ngữ HTML 49
2.1.2 Cấu trúc trang HTML 50
2.1.3 Chú thích trong HTML 50
2.1.4 Tạo tài liệu HTML 50
2.2 PHẦN TỬ HTML **52**
2.2.1 Phần tử HTML là gì 52
2.2.2 Thẻ HTML 52

2.2.3 Thuộc tính HTML 53
2.3 QUY ƯỚC KHI VIẾT MÃ HTML 55
2.3.1 Một số quy ước quan trọng 55
2.3.2 Kiểm tra mã HTML 55
2.4 PHẦN ĐẦU (HEAD) CỦA TÀI LIỆU 56
2.4.1 Tiêu đề 57
2.4.2 Ảnh biểu tượng 57
2.4.3 Các thẻ meta 58
2.4.4 Đường dẫn cơ sở 59
2.5 PHẦN TỬ KHỐI VÀ PHẦN TỬ NỘI TUYẾN 59
2.5.1 Phần tử khối 59
2.5.2 Phần tử nội tuyến 60
2.6 PHẦN TỬ NGỮ NGHĨA 60
2.7 VĂN BẢN 65
2.7.1 Tiêu đề 65
2.7.2 Đoạn văn 67
2.7.3 Định dạng văn bản 69
2.7.4 Ngắt dòng 71
2.7.5 Văn bản mã lệnh (code) 71
2.8 BẢNG BIỂU 72
2.8.1 Thẻ Table 72
2.8.2 Thuộc tính colspan và rowspan 73
2.8.3 Tiêu đề cho bảng 73
2.8.4 Đầu, thân và chân bảng trong HTML 74
2.9 DANH SÁCH 75
2.9.1 Danh sách không có thứ tự 75
2.9.2 Danh sách có thứ tự 76
2.9.3 Danh sách định nghĩa 78
2.10 LIÊN KẾT 78
2.10.1 Tạo liên kết 78
2.10.2 Liên kết cục bộ 79
2.10.3 Thuộc tính target 79
2.10.4 Hình ảnh liên kết 79
2.10.5 Tiêu đề liên kết 79
2.10.6 Liên kết đánh dấu 79
2.11 BIỂU MẪU 80
2.11.1 Biểu mẫu là gì? 80
2.11.2 Ô nhập văn bản 82
2.11.3 Vùng nhập văn bản 85
2.11.4 Lựa chọn 85
2.11.5 Nhập thời gian 88
2.11.6 Nhập tập tin 91
2.11.7 Nút bấm 91
2.11.8 Các phần tử biểu mẫu khác 93
2.11.9 Các thuộc tính áp dụng cho phần tử nhập 94
2.12 NỘI DUNG ĐA PHƯƠNG TIỆN 97
2.12.1 Hình ảnh Bitmap 97
2.12.2 Hình ảnh vector 98

2.12.3 Âm thanh và phim 99
2.12.4 Tập tin PDF 101
THỰC HÀNH 102
CÂU HỎI ÔN TẬP LÝ THUYẾT 102
BÀI TẬP TỰ THỰC HÀNH 103
Bài 1. Tạo favicon 103
Bài 2. Hãy tạo một giao diện như sau với các ảnh là svg 105
Bài 3. Tạo form 106
TÀI LIỆU THAM KHẢO 106
CHƯƠNG 3: CÁC THUỘC TÍNH ĐỊNH DẠNG CƠ BẢN CỦA CSS 107
3.1 GIỚI THIỆU 107
3.1.1 CSS là gì? 107
3.1.2 Cú pháp CSS 107
3.1.3 Ghi chú thích cho CSS 107
3.2 CHÈN CSS VÀO TÀI LIỆU HTML 108
3.2.1 CSS bên ngoài 108
3.2.2 CSS nội bộ 109
3.2.3 CSS nội tuyến 109
3.2.4 Kết hợp CSS 110
3.2.5 Khai báo !important 111
3.2.6 Thuộc tính all 111
3.3 CÁC LOẠI BỘ CHỌN 112
3.3.1 Bộ chọn CSS 112
3.3.2 Bộ chọn đơn giản 112
3.3.3 Bộ chọn thuộc tính 115
3.3.4 Bộ chọn tổ hợp 116
3.3.5 Bộ chọn lớp giả 118
3.3.6 Bộ chọn phần tử giả 121
3.4 SỬ DỤNG MÀU SẮC 122
3.4.1 Màu hệ thập lục 122
3.4.2 RGB 123
3.4.3 HSL 123
3.4.4 Tên màu 124
3.4.5 Độ trong suốt 124
3.5 ĐỊNH DẠNG NỀN 125
3.5.1 Màu nền 125
3.5.2 Hình nền 125
3.5.3 Vị trí hình nền 126
3.5.4 Hình nền cố định 127
3.5.5 Cách viết rút gọn 127
3.6 ĐỊNH DẠNG VIỀN 127
3.6.1 Kiểu viền 127
3.6.2 Độ rộng đường viền CSS 128
3.6.3 Màu CSS 128
3.6.4 Border từng phía 128
3.6.5 Cách viết rút gọn 129
3.6.6 Viền bo tròn 129
3.7 ĐỊNH DẠNG VĂN BẢN 129

3.7.1 Màu sắc 129
3.7.2 Căn chỉnh văn bản 129
3.7.3 Trang trí văn bản 130
3.7.4 Chuyển đổi văn bản 130
3.7.5 Thụt đầu dòng 130
3.7.6 Khoảng cách chữ 131
3.7.7 Khoảng cách từ 131
3.7.8 Chiều cao giữa các dòng 131
3.7.9 Hướng văn bản 131
3.7.10 Bóng văn bản 131
3.7.11 Font chữ 132
3.8 ĐỊNH DẠNG LIÊN KẾT 134
3.9 DANH SÁCH 135
3.10 BẢNG 137
3.11 BIỂU MẪU 138
3.12 SỬ DỤNG BIẾN 142
THỰC HÀNH 142
CÂU HỎI ÔN TẬP LÝ THUYẾT 142
Bài 1. Sử dụng màu sắc 144
Bài 2. Tạo và định dạng bảng, văn bản 144
Bài 3. Sửa form đã thiết kế 144
Bài 4. Tạo form đăng nhập 145
TÀI LIỆU THAM KHẢO 145
CHƯƠNG 4: THIẾT KẾ BỐ CỤC 146
4.1 ĐỘ RỘNG VÀ CHIỀU CAO 146
4.1.1 Thuộc tính width và height 146
4.1.2 Chiều rộng và chiều cao tối đa 147
4.2 ĐƠN VỊ KÍCH THƯỚC 147
4.2.1 Các loại đơn vị đo kích thước 147
4.2.2 Đơn vị tuyệt đối 148
4.2.3 Đơn vị tương đối 148
4.3 THUỘC TÍNH MARGIN 149
4.3.1 Margin là gì 149
4.3.2 Cách viết rút gọn 150
4.3.3 Giá trị auto 151
4.3.4 Giá trị inherit 152
4.3.5 Gom lề 153
4.4 THUỘC TÍNH PADDING 153
4.4.1 Padding là gì 153
4.4.2 Cách viết rút gọn 154
4.5 MÔ HÌNH HỘP 154
4.6 CĂN CHỈNH 156
4.6.1 Căn giữa cho phần tử 156
4.6.2 Căn chỉnh văn bản 157
4.6.3 Căn trái phải cho phần tử 157
4.6.4 Căn giữa theo chiều dọc 158
4.7 THIẾT LẬP VỊ TRÍ 160
4.7.1 static 161

4.7.2 relative 161
4.7.3 fixed 162
4.7.4 absolute 163
4.7.5 sticky 164
4.7.6 z-index 165
4.8 THUỘC TÍNH TRÀN OVERFLOW 166
4.9 THUỘC TÍNH TRÔI FLOAT 168
4.10 THUỘC TÍNH DISPLAY 171
4.11 HỘP LINH HOẠT (FLEXBOX) 173
4.11.1 Flexbox tạo ra các bố cục linh hoạt 173
4.11.2 Hộp chứa (flex container) 174
4.11.3 Các phần tử flex item 179
4.12 LƯỚI (GRID) 182
4.12.1 Khái niệm lưới (Grid) 182
4.12.2 Grid Container 183
4.12.3 Các phần tử grid Item 185
4.13 BỐ CỤC WEBSITE 188
4.13.1 Bố cục Website 188
4.13.2 Header 188
4.13.3 Thanh điều hướng 189
4.13.4 Content 189
4.13.5 Footer 190
4.14 THIẾT KẾ WEB ĐÁP ỨNG 190
4.14.1 Thiết kế Web đáp ứng là gì? 190
4.14.2 Viewport 191
4.14.3 Media Queries 192
4.14.4 Thiết kế "Mobile First" 194
4.14.5 Responsive Images & Videos 195
4.14.6 Responsive Menu 197
4.14.7 Ẩn các phần tử không cần thiết 198
4.14.8 Cỡ chữ 198
THỰC HÀNH 199
Bài 1. Thiết kế Website 1 199
Bài 2. Thiết kế Website 2 204
Bài 3. Thiết kế Website MinhBlogs 207
CÂU HỎI ÔN TẬP LÝ THUYẾT 224
BÀI TẬP TỰ THỰC HÀNH 227
Bài 1. Thiết kế biểu mẫu đáp ứng 227
Bài 2. Thiết kế Website với 3 kích thước cho 3 thiết bị 227
Bài 3. Sử dụng Grid trong thiết kế bố cục 227
TÀI LIỆU THAM KHẢO 227
CHƯƠNG 5: LẬP TRÌNH JAVASCRIPT CĂN BẢN 229
5.1 TỔNG QUAN VỀ JAVASCRIPT 229
5.1.1 Khái niệm JavaScript 229
5.1.2 Chèn JavaScript vào tài liệu HTML 230
5.1.3 Hiển thị log 230
5.1.4 Ghi chú thích 231
5.1.5 Từ khóa, từ dành riêng 232

5.1.6 Thực thi mã Javascript bằng nodejs 233
5.2 SỬ DỤNG BIẾN 233
5.2.1 Khai báo biến 233
5.2.2 Phạm vi biến 234
5.2.3 Cẩu - Hoisting 235
5.3 KIỂU DỮ LIỆU 235
5.3.1 Kiểu số 236
5.3.2 Kiểu chuỗi ký tự 237
5.3.3 Kiểu boolean 239
5.3.4 Kiểu đối tượng 239
5.3.5 Kiểu rỗng null 240
5.3.6 Kiểm tra kiểu 240
5.3.7 Chuyển kiểu 241
5.4 TOÁN TỬ 241
5.4.1 Toán tử gán 242
5.4.2 Toán tử toán học 242
5.4.3 Toán tử so sánh 242
5.4.4 Toán tử bitwise 242
5.5 HÀM 243
5.5.1 Khai báo và sử dụng hàm 243
5.5.2 Hàm mũi tên 243
5.5.3 Các hàm của các đối tượng có sẵn 244
5.6 MẢNG 244
5.6.1 Mảng được sử dụng để lưu trữ nhiều giá trị vào một biến 244
5.6.2 Phương thức 245
5.6.3 Sắp xếp mảng 246
5.6.4 Duyệt mảng 246
5.7 CÂU ĐIỀU KIỆN 247
5.7.1 Câu lệnh if else 247
5.7.2 Câu lệnh switch 248
5.8 VÒNG LẶP 249
5.8.1 Vòng lặp while 249
5.8.2 Vòng lặp do while 250
- Vòng lặp for 251
5.8.3 Câu lệnh break và continue 253
5.9 XỬ LÝ LỖI 253
5.9.1 try catch 253
5.9.2 Ném lỗi 254
5.9.3 Lệnh finally 254
5.10 ĐỐI TƯỢNG 255
5.10.1 Tạo đối tượng bằng {} 255
5.10.2 Tạo đối tượng bằng phương thức khởi tạo 256
5.10.3 Tạo đối tượng bằng new [class Name] 257
5.10.4 Tính chất kế thừa 257
5.10.5 Phương thức tĩnh 258
5.10.6 Phương thức getter và setter 259
CÂU HỎI ÔN TẬP LÝ THUYẾT 259
TÀI LIỆU THAM KHẢO 261

CHƯƠNG 6: JAVASCRIPT TRONG THIẾT KẾ WEBSITE 262
6.1 ĐỐI TƯỢNG CỬA SỔ 262
6.1.1 Đối tượng cửa sổ là gì? 262
6.1.2 Các phương thức quan trọng 263
6.1.3 Các thuộc tính chứa các đối tượng quan trọng 267
6.2 ĐỐI TƯỢNG TÀI LIỆU 271
6.2.1 Tìm phần tử HTML 272
6.2.2 Thay đổi thuộc tính 273
6.3 XỬ LÝ SỰ KIỆN 273
6.3.1 Gắn sự kiện vào DOM 273
6.3.2 Lan truyền sự kiện 275
6.4 NÚT (NODE) 276
6.4.1 Mối quan hệ nút 276
6.4.2 Thuộc tính nodeName 277
6.4.3 Thuộc tính nodeValue 277
6.4.4 Thuộc tính nodeType 278
6.4.5 Tạo các phần tử HTML mới (Nút) 278
6.4.6 Xóa các phần tử HTML hiện có 278
6.5 XÁC THỰC BIỂU MẪU 279
6.6 KÉO/ THẢ 280
6.6.1 Tạo một phần tử có thể kéo 280
6.6.2 Kéo - ondragstart và setData() 281
6.6.3 ondragover 281
6.6.4 Thả - ondrop 281
6.7 CANVAS 282
6.8 JSON 283
6.8.1 JSON là gì? 283
6.8.2 JSON và XML 284
6.8.3 Kiểu dữ liệu JSON 285
6.9 AJAX 286
6.9.1 AJAX là gì? 286
6.9.2 Các bước để thực hiện AJAX 287
6.9.3 Các thuộc tính và phương thức 289
THỰC HÀNH 290
Bài 1. Tạo một đối tượng chuyển động: 290
Bài 2. Bài thức hành với cơ sở dữ liệu JSON trên Firebase 291
Bài 3. Sử dụng PostMan 294
Bài 4. Thực hành thêm xóa dữ liệu trên Firebase 296
Bài 5. Đọc và lưu dữ liệu vào Minh Blogs 297
CÂU HỎI ÔN TẬP LÝ THUYẾT 299
BÀI TẬP TỰ THỰC HÀNH 302
Bài 1. Xử lý biểu mẫu 302
Bài 2. Website giới thiệu sản phẩm 302
Bài 3. Tạo một trò chơi đơn giản 303
TÀI LIỆU THAM KHẢO 303
CHƯƠNG 7: CÁC CÔNG CỤ HỖ TRỢ 304
7.1 EMMET 304
7.2 GIT AND GITHUB 306

7.3 SASS 307
7.3.1 Giới thiệu 307
7.3.2 Cách sử dụng Sass để lồng các quy tắc kiểu 308
7.3.3 Sử dụng biến và mixins của Sass 309
7.4 BOOTSTRAP 310
7.4.1 Giới thiệu 310
7.4.2 Cách sử dụng hệ thống lưới Bootstrap 311
7.4.3 Cách sử dụng các thành phần Bootstrap 313
7.4.4 Biểu tượng (icon) 313
7.5 JQUERY 314
7.5.1 $().ready() 316
7.5.2 Các bộ chọn 316
7.5.3 Lấy và thiết lập giá trị 317
7.5.4 Tìm kiếm phần tử xung quanh 319
7.5.5 Định dạng kiểu 320
7.5.6 Sự kiện 321
7.5.7 Hiệu ứng 321
7.5.8 Sử dụng Ajax 323
7.5.9 Các chương trình cài cắm 326
THỰC HÀNH 326
CÂU HỎI ÔN TẬP LÝ THUYẾT 329
BÀI TẬP TỰ THỰC HÀNH 332
TÀI LIỆU THAM KHẢO 332
CHƯƠNG 8: TRIỂN KHAI WEBSITE 333
8.1 LỰA CHỌN TÊN MIỀN (DOMAIN) 333
8.1.1 Tên miền (domain) 333
8.1.2 Lựa chọn nơi mua tên miền 334
8.2 LỰA CHỌN NƠI ĐẶT WEBSITE (WEB HOST) 334
8.2.1 Dịch vụ lưu trữ Website (Web host) 334
8.2.2 Lựa chọn nơi mua Web host 335
8.2.3 Chuyển Website lên Web host 335
8.3 MỘT SỐ KỸ NĂNG CẦN THIẾT KHÁC KHI TRIỂN KHAI WEB 336
8.3.1 Kiểm thử Website sau khi tải lên Web host 336
8.3.2 Khai báo Website tới các máy tìm kiếm 338
8.3.3 Kiểm soát các trang được đánh chỉ mục và tìm kiếm 343
8.3.4 Bảo trì Website 344
THỰC HÀNH 347
Bài 1. Mua tên miền 347
Bài 2. Mua Web host 349
Bài 3. Chuyển mã nguồn trang Web lên Web host 351
Bài 4. Triển khai Website lên Firebase 352
CÂU HỎI ÔN TẬP LÝ THUYẾT 352
BÀI TẬP TỰ THỰC HÀNH 355
TÀI LIỆU THAM KHẢO 355
PHỤ LỤC 356
PHỤ LỤC 1: BẢNG CÁC PHẦN TỬ HTML 356
PHỤ LỤC 2: BẢNG TRA CỨU THUỘC TÍNH CSS 358
PHỤ LỤC 3: BẢNG DANH MỤC TỪ DÀNH RIÊNG TRONG JAVASCRIPT 360

TÀI LIỆU THAM KHẢO 361

DANH MỤC TỪ VIẾT TẮT

Từ viết tắt	Tiếng Anh	Tiếng Việt
API	Application Programming Interface	Giao diện lập trình ứng dụng
CSS	Cascading Style Sheets	Ngôn ngữ được dùng để định dạng phần tử HTML
DOM	Document Object Model	Mô hình đối tượng tài liệu
HTML	Hypertext Markup Language	Ngôn ngữ đánh dấu siêu văn bản
IANA	Internet Assigned Numbers Authority	Tổ chức cấp phát số hiệu Internet
IDE	Integrated Development Environment	Môi trường phát triển tích hợp
SEO	Search Engine Optimization	Tối ưu hóa công cụ tìm kiếm
TLD	Top Level Domain	Tên miền cấp cao nhất
URL	Uniform Resource Locator	Hệ thống định vị tài nguyên thống nhất
VPS	Virtual Private Server	Máy chủ riêng ảo
VSC	Visual Studio Code	Công cụ hỗ trợ lập trình được phát triển bởi Microsoft
RWD	Responsive Web Design	Thiết kế Web đáp ứng

DANH MỤC HÌNH

Hình 1-1 Kết nối giữa các thiết bị trên mạng Internet 21
Hình 1-3 Một trang Web tĩnh 22
Hình 1-4 Cách thức làm việc của một trang Web tĩnh 22
Hình 1-5 Cách thức làm việc của một trang Web động 23
Hình 1-6 Ví dụ một trang Web động 24
Hình 8-1 Các thành phần của đường dẫn Web 25
Hình 1-7 Thiết kế giao diện người dùng (Front-end) 26
Hình 1-8. Lập trình xử lý phía máy chủ (Back-end) 27
Hình 1-9 Quy trình phát triển một Website 27
Hình 1-10 Ví dụ về sơ đồ trang Web 28
Hình 1-11 Ví dụ một phác thảo được tạo bởi công cụ Mogups 29
Hình 1-12 Thiết kế giao diện (UI), trải nghiệm (UX) cho Website 29
Hình 1-13 Một số loại kiểm thử 31
Hình 1-14 Thống kê thị phần trình duyệt ..... **Error! Bookmark not defined.**
Hình 1-15 Kiểm tra sự tương thích của trang Web trên nhiều loại trình duyệt 32
Hình 1-16 Thiết kế Web đáp ứng trên nhiều loại thiết bị 34
Hình 1-17 Kiểm tra Website hoạt động trên nhiều loại kích thước màn hình 35
Hình 1-19 Một tài liệu HTML đơn giản được hiển thị bởi trình duyệt 37
Hình 1-20 Bảng hỗ trợ thuộc tính CSS của mỗi trình duyệt 38
Hình 1-21 Một tài liệu HTML được áp dụng CSS 39
Hình 1-22 Cách thức một file JavaScript được tải về và thực hiện 40
Hình 1-23 Biểu tượng bộ 3 công cụ của Adobe 41
Hình 1-24 Công cụ Figma 41
Hình 1-25 Công cụ Visual Studio Code 42
Hình 1-26 Công cụ Sublime Text 42
Hình 1-27 Công cụ IntelliJ IDEA của JetBrains 43
Hình 1-28 Công cụ CodePen.io 44
Hình 1-29 Công cụ CodeSanbox.io 44
Hình 1-30 Công cụ jsFiddle 45
Hình 1-31 Xem mã nguồn Website trên nhiều trình duyệt khác nhau 46
Hình 1-32 Công cụ Developer Tools của trình duyệt Chrome 46
Hình 2-1 Cấu trúc một tài liệu HTML 50
Hình 2-2 Tạo và biên tập file HTML treenVisual Studio Code 51

Hình 2-3 Cài đặt chức năng mở rộng “Live Preview” trên Visual Studio Code 51
Hình 2-4 Xem trước nội dung Website bằng “Live Preview” 52
Hình 2-5 Phần tử HTML 52
Hình 2-6 Công cụ kiểm tra mã HTML 56
Hình 2-7 Các lỗi được phát hiện ra bởi công cụ Validator 56
Hình 2-8 Tiêu đề sẽ được xuất hiện trên các Tab của trình duyệt 57
Hình 2-9 Favicon xuất hiện ở cửa sổ bookmark 57
Hình 2-10 Các favicon cạnh tiêu đề trên cửa sổ Tab của trình duyệt trên Android 58
Hình 2-11 Một <section> thể hiện thông tin liên hệ 62
Hình 2-12 Tiêu đề từ h1 đến 6 khi hiển thị trên trình duyệt 66
Hình 2-13 Ví dụ sử dụng tiêu đề <h1> và <h2> trên một bài viết 67
Hình 2-14 Phần tử <q> dùng để hiển thị một câu nói hoặc trích dẫn 68
Hình 2-15 Sử dụng phần tử <abbr> để mô tả một từ viết tắt 69
Hình 2-16 Hiển thị mã lệnh trên trang Web 71
Hình 2-17 Danh sách không có thứ tự với các dấu chấm đầu dòng 75
Hình 2-18 Danh sách với các dấu vuông đầu dòng 76
Hình 2-19 Một danh mục được tạo bởi phần tử <dl>, <dt>, và <dd> 78
Hình 2-20 Ví dụ một biểu mẫu đăng ký khóa học 80
Hình 2-21 Ô nhập văn bản 82
Hình 2-22 Ô nhập địa chỉ hòm thư 82
Hình 2-23 Ô nhập số 83
Hình 2-24 Điều khiển để chọn giá trị số 83
Hình 2-25 Ô nhập tìm kiếm 83
Hình 2-26 Ô nhập số điện thoại 84
Hình 2-27 Ô nhập mật khẩu 84
Hình 2-28 Ô nhập địa chỉ URL 85
Hình 2-29 Vùng nhập văn bản 85
Hình 2-30 Ô lựa chọn Radio 86
Hình 2-31 Danh sách lựa chọn 86
Hình 2-32 Ô lựa chọn checkbox 87
Hình 2-33 Các nhóm lựa chọn 88
Hình 2-34 Ô lựa chọn ngày giờ 89
Hình 2-35 Ô lựa chọn tháng 89
Hình 2-36 Ô lựa chọn tuần 90
Hình 2-37 Ô lựa chọn giờ 90

Hình 2-38 Ô lựa chọn tập tin 91
Hình 2-39 Nhóm các phần tử nhập bằng cách sử dụng <fieldset> 93
Hình 2-40 Phần tử hiển thị tiến độ <process> 94
Hình 2-41 Hiển thị kết quả đầu ra bằng <output> 94
Hình 2-42 Kết quả hiển thị một hình ảnh 98
Hình 2-43 Hình ảnh vector được tạo ra 99
Hình 2-44 Một video với phụ đề kèm theo 101
Hình 3-1 Cú pháp của một khai báo CSS 107
Hình 3-2 Kết quả sau khi áp dụng css bên ngoài 109
Hình 3-3 Kết quả khi áp dụng CSS nội tuyến 110
Hình 3-4 Bức ảnh được định danh bằng id áp dụng CSS 113
Hình 3-5 Áp dụng CSS trên các phần tử có class là readmore 113
Hình 3-6 Minh họa cách làm việc của bộ chọn anh em liền kề (dấu >) 117
Hình 3-7 Minh họa cách làm việc của bộ chọn anh em liền kề (dấu +) 118
Hình 3-8 Minh họa cách làm việc của bộ chọn anh em (dấu +) 118
Hình 3-9 Thanh điều hướng được tạo ra bởi CSS 119
Hình 3-10 Bảng một số màu sử dụng mã màu thập lục 123
Hình 3-11 Màu sắc và tên màu 124
Hình 3-12 Hình ảnh với opacity tương ứng là 1 và 0.75 124
Hình 3-13 Biểu mẫu chưa được định dạng và đã được định dạng bởi CSS 139
Hình 3-14 Định dangk các nút bấm sau khi áp dụng CSS 140
Hình 4-1 Thiết lập chiều cao và chiều rộng cho một phần tử khối 147
Hình 4-2 Cấu hình cỡ chữ sử dụng đơn vị tương đối 149
Hình 4-3 Phần tử <body> sau khi bỏ margin 150
Hình 4-4 Xem kích thước, padding và margin bằng trình duyệt 151
Hình 4-5 Sử dụng margin:auto để căn giữa một phần tử khối 152
Hình 4-6 Phần tử <p> kế thừa giá trị margin từ phần tử cha 153
Hình 4-7 Căn chỉnh giữa cho phần tử khối 157
Hình 4-8 Sử dụng float để căn phải một phần tử 158
Hình 4-9 Căn giữa theo chiều dọc 159
Hình 4-10 Hai phần tử nằm liền kề nhau 161
Hình 4-11 Div1 được thiết lập position:relative 162
Hình 4-12 Div1 được thiết lập position:fixed 163
Hình 4-13 Div1 được thiết lập position:absolute 164
Hình 4-14 Div1 được thiết lập position:fixx 165

Hình 4-15 Div nằm dưới vì có z-index nhỏ hơn 166
Hình 4-16 Nội dung bị tràn do dữ liệu bên trong cần nhiều không gian để hiển thị 167
Hình 4-17 Trường hợp div không sử dụng float 169
Hình 4-18 Trường hợp div được thiết lập float:left 170
Hình 4-19 Các phần tử <li> với thiết lập display mặc định 172
Hình 4-20 Các phần tử <li> với thiết lập display:inline 173
Hình 4-21 Hoạt động của flexbox dựa trên việc cấu hình Container và Item 174
Hình 4-22 Một số thuật ngữ cần biết khi làm việc với Flexbox 175
Hình 4-23 Thuộc tính flex-direction 175
Hình 4-24 Thuộc tính flex-wrap 176
Hình 4-25 Thuộc tính align-items 177
Hình 4-26 Thuộc tính justify-content 178
Hình 4-27 Thuộc tính align-content 179
Hình 4-28 Thuộc tính align-self 181
Hình 5-1 Phần tử a sau khi thay đổi nội dung và thuộc tính bằng Javascript 229
Hình 5-2 Kết quả hiển thị Log trên cửa sổ Console 231
Hình 6-1 Thông tin về đối tượng window 262
Hình 6-2 Hộp thoại thông báo 264
Hình 6-3 Hộp thoại xác nhận 264
Hình 6-4 Hộp thoại nhập 265
Hình 6-5 Bật Popup trên trình duyệt Chrome 266
Hình 6-6 Tùy chọn Print của trình duyệt 267
Hình 6-7 Hiển thị kích thước khả dụng của màn hình 268
Hình 6-8 Các thuộc tính giá trị của đối tượng location 268
Hình 6-9 Thanh điều hướng trên trình duyệt 269
Hình 6-10 Yêu cầu truy cập vị trí của người dùng 269
Hình 6-11 Xem cookie bằng trình duyệt 270
Hình 7-1 Các công cụ có thể cài Emmet 305
Hình 7-2 Giao diện của Github Desktop 307
Hình 7-3 Danh sách in ra bởi jQuery 316

DANH MỤC BẢNG

Bảng 1-1 Thống kê thị phần các trình duyệt (tháng 10 năm 2020)..... 25
Bảng 2-1 Mô tả các thuộc tính tiêu đề từ h1 đến h6..... 65
Bảng 2-2 Bảng thuộc tính hỗ trợ cho phần tử audio/video 100
Bảng 3-1 Danh sách các bộ chọn lớp giả khác..... 120
Bảng 5-1 Danh sách từ khóa trong JavaScript 232
Bảng 7-1 Các thao tác phổ biến trong Emmet..... 305
Bảng 7-2 Bảng cú pháp viết tắt trong Emmet 305

LỜI NÓI ĐẦU

Hiện nay, việc ứng dụng mạng Internet vào các hoạt động trong cuộc sống, sản xuất kinh doanh, quảng bá sản phẩm ngày càng trở lên phổ biến. Nó cho phép kết nối các thiết bị trên thế giới và giúp con người dễ dàng tìm kiếm, truy cập thông tin thông qua các Website. Theo thống kê sơ bộ của Internet Live Stats hiện nay trên thế giới có khoảng hơn 500 triệu Website đang hoạt động. Nó mang lại lượng thông tin khổng lồ cho người dùng và hỗ trợ hiệu quả cho hoạt động quảng bá thông tin, quảng cáo sản phẩm, dịch vụ kinh doanh và được xem là cánh tay đắc lực cho chiến lược Marketing của mỗi tổ chức, doanh nghiệp. Chính vì vậy, vai trò của Website đối với tổ chức, doanh nghiệp là rất quan trọng trong mọi lĩnh vực cho sự tồn tại và phát triển ở thời điểm hiện tại cũng như tương lai. Nhu cầu thiết kế, xây dựng, vận hành một hệ thống Website hiện rất lớn. Đó cũng là cơ hội rất tốt để các bạn sinh viên tốt nghiệp có thể tìm được công việc thuận lợi trong lĩnh vực phát triển ứng dụng Web.

Giáo trình Thiết kế Web được biên soạn bám sát theo chương trình khung của đề cương chi tiết học phần Thiết kế Web do Trường Đại học Kinh tế Quốc dân ban hành, được sử dụng cho sinh viên hệ đại học chính quy ngành Công nghệ thông tin và Hệ thống thông tin quản lý. Giáo trình gồm 8 chương và phụ lục sẽ cung cấp các kiến thức chuyên sâu về ngôn ngữ siêu văn bản đánh dấu (HTML), định dạng HTML với CSS, ngôn ngữ lập trình JavaScript. Trong mỗi chương, nhóm tác giả có đưa những ví dụ minh hoạ, những câu hỏi và bài tập thực hành một cách có hệ thống để giúp sinh viên nắm bắt được lý thuyết và tiếp cận được với kiến thức thực tiễn trong thiết kế Web.

Chương 1: Giới thiệu nguyên lý hoạt động của Website, các khái niệm căn bản liên quan tới một số thành phần cơ bản giúp tạo ra một trang Web. Bên cạnh đó, quy trình phát triển một trang Web và một số công cụ cần thiết để bắt đầu thực hiện phát triển một trang Web cũng sẽ được cung cấp tới người học. Khi hoàn thành chương này, người học sẽ có nền tảng cần thiết để học cách xây dựng Website.

Chương 2: Giới thiệu về ngôn ngữ đánh dấu siêu văn bản (HTML), các phần tử cơ bản của ngôn ngữ HTML. Bên cạnh đó, người học sẽ được học cách sử dụng các phần tử HTML cơ bản để xây dựng nội dung và cấu trúc của một trang Web.

Chương 3: Giới thiệu về ngôn ngữ CSS - một ngôn ngữ được dùng để định dạng các phần tử do ngôn ngữ đánh dấu văn bản tạo ra. Người học sẽ được học các thuộc tính cơ bản trong ngôn ngữ CSS. Bên cạnh đó, cách nhúng CSS nhằm kiểm soát bố cục và tạo ra phong cách thống nhất cho một trang Web cũng sẽ được trình bày.

Chương 4: Giới thiệu cách sử dụng CSS để kiểm soát bố cục của một trang Web. Điều đó có nghĩa là người học có thể kiểm soát vị trí của mỗi phần tử HTML xuất hiện trên trang Web. Sau khi học xong chương này, người học có thể triển khai các bố cục mong muốn liên quan đến một trang Web. Ngoài ra, một công nghệ tiên tiến trong thiết kế Web là Web đáp ứng cũng được đề cập nhằm giúp một trang Web có thể hiển thị tối ưu trên các thiết bị có độ rộng màn hình khác nhau.

Chương 5: Giới thiệu các khái niệm căn bản về JavaScript, một trong những thành phần quan trọng trong phát triển ứng dụng Web. Sau khi hoàn thành chương này, người học có thể biết

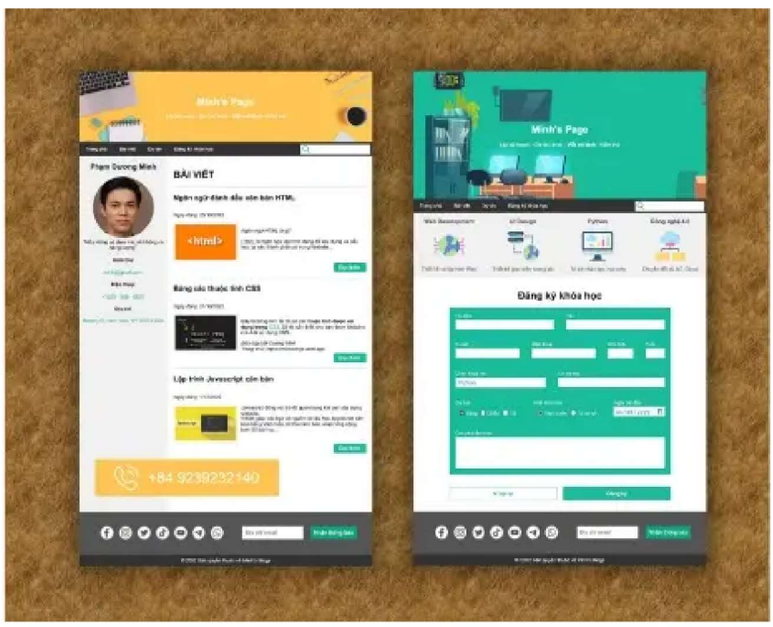
cách sử dụng các câu lệnh căn bản trong JavaScript, nhúng các câu lệnh JavaScript trong một trang Web bất kỳ.

Chương 6: Giới thiệu về đối tượng tài liệu (DOM) nhằm thực hiện việc ghi nhận, thay đổi, thêm hoặc xóa các phần tử trong HTML. Bên cạnh đó, người học sẽ được tìm hiểu cách sử dụng JavaScript để xử lý các sự kiện, xác thực các biểu mẫu trong một trang Web. Bên cạnh đó, các kiểu dữ liệu (JSON, XML) & cách thức tải dữ liệu từ các hệ thống Website khác không cần tải lại trang (AJAX) cũng sẽ được đề cập.

Chương 7: Bên cạnh HTML và CSS, các nhà phát triển Web thường sử dụng các công cụ phát triển Web được cung cấp bởi bên thứ ba nhằm tăng tính hiệu quả cũng như năng suất trong công việc. Các công cụ này có thể là một plugin giúp tạo mã HTML tự động từ các từ viết tắt, đến một framework cho phép phát triển Web được nhanh chóng hơn. Ngoài ra, người học cũng được tìm hiểu một số công cụ của bên thứ ba phổ biến nhằm xây dựng một trang Web. Đây cũng sẽ là cơ sở để người học có thể đưa ra những lựa chọn phù hợp về công nghệ cũng như giải pháp cho việc phát triển Web trong tương lai.

Chương 8: Hướng dẫn cách thức đưa một trang Web từ máy tính cục bộ lên môi trường mạng Internet. Để làm điều đó, người học sẽ làm quen với các khái niệm như tên miền (Domain), nơi lưu trữ trang Web trên không gian mạng Internet (Web host), ứng dụng giúp chuyển các file mã nguồn trang Web từ máy tính cục bộ đến một máy chủ Web có kết nối tới mạng Internet (FTP client). Bên cạnh đó, người học cũng được trang bị một số kinh nghiệm trong việc kiểm thử, tối ưu hoá việc vận hành Website trên môi trường trực tuyến.

Xuyên suốt toàn bộ giáo trình gắn vào một bài hướng dẫn cụ thể là thiết kế một Website cá nhân. Website có các giao diện gồm các bài viết hướng dẫn học các kiến thức về Công nghệ thông tin và biểu mẫu đăng ký khóa học. Toàn bộ giáo trình sẽ hướng dẫn theo một quy trình đầy đủ để tạo ra Website này từ việc phác thảo, viết mã cho đến việc triển khai và kiểm thử cho Website. Giao diện của Website sau khi hoàn thành sẽ như sau:



Giáo trình Thiết kế Web do TS. Phạm Xuân Lâm làm chủ biên. Tập thể tác giả bao gồm: ThS. Phạm Đức Trung, ThS. Cao Thị Thu Hương, ThS. Chu Văn Huy.

Dù đã rất cố gắng nhưng chắc chắn cuốn tài liệu này còn có những sai sót. Rất mong nhận được sự góp ý kiến của các đồng nghiệp, các em sinh viên để cuốn giáo trình này ngày càng hoàn thiện hơn.

Các ý kiến đóng góp xin gửi về theo địa chỉ:

- *P1310, Bộ môn Công nghệ thông tin, Viện Công nghệ thông tin & Kinh tế số, Trường Đại học Kinh tế Quốc dân*
- *Điện thoại: 0937.638683*
- *Email: lampx@neu.edu.vn*

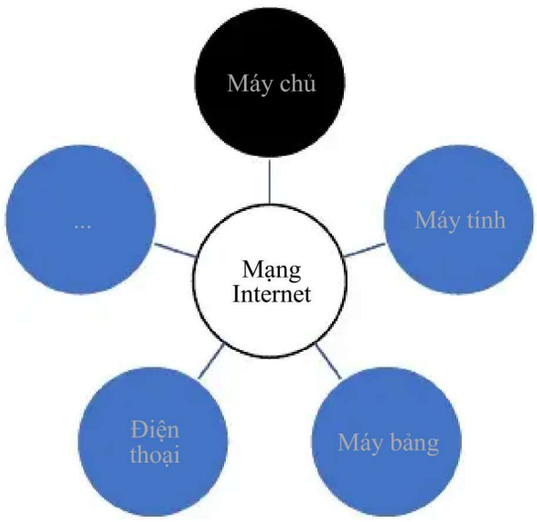
CHƯƠNG 1: GIỚI THIỆU VỀ THIẾT KẾ WEBSITE

Chương này giới thiệu nguyên lý hoạt động của Website, các khái niệm căn bản liên quan tới một số thành phần cơ bản giúp tạo ra một trang Web. Bên cạnh đó, quy trình phát triển một trang Web và một số công cụ cần thiết để bắt đầu thực hiện phát triển một trang Web cũng sẽ được cung cấp tới người học. Khi hoàn thành chương này, người học sẽ có nền tảng cần thiết để học cách xây dựng Website.

1.1 TỔNG QUAN VỀ THIẾT KẾ WEBSITE

1.1.1 Website là gì?

Website là tập hợp các trang thông tin (trang Web) được tạo ra để hiển thị nội dung trên mạng Internet. Một trang Web có thể chứa nhiều loại nội dung, bao gồm văn bản, hình ảnh, video, âm thanh và các liên kết đến các trang Web khác. Mỗi trang Web được định danh bằng một địa chỉ duy nhất gọi là URL (Uniform Resource Locator), cho phép người dùng truy cập vào nội dung của trang Web thông qua **trình duyệt Web** (Web browser) của họ. Các trang Web có thể được tạo ra cho mục đích cá nhân, doanh nghiệp, giáo dục hoặc giải trí. Website được tạo ra đặt trên **máy chủ** (Server), được truy cập bởi **máy khách** (Client) thông qua trình duyệt. Trong đó máy khách có thể là các máy tính, máy bảng, điện thoại là những thiết bị sử dụng Website. Máy chủ chứa thông tin, dữ liệu của Website. Người dùng có thể yêu cầu các thông tin dữ liệu và nhận các dữ liệu này. Sơ đồ thể hiện mối quan hệ của một Website tiêu biểu như sau:



Hình 1-1 Kết nối giữa máy khách và máy chủ thông qua mạng Internet

1.1.2 Mạng máy tính

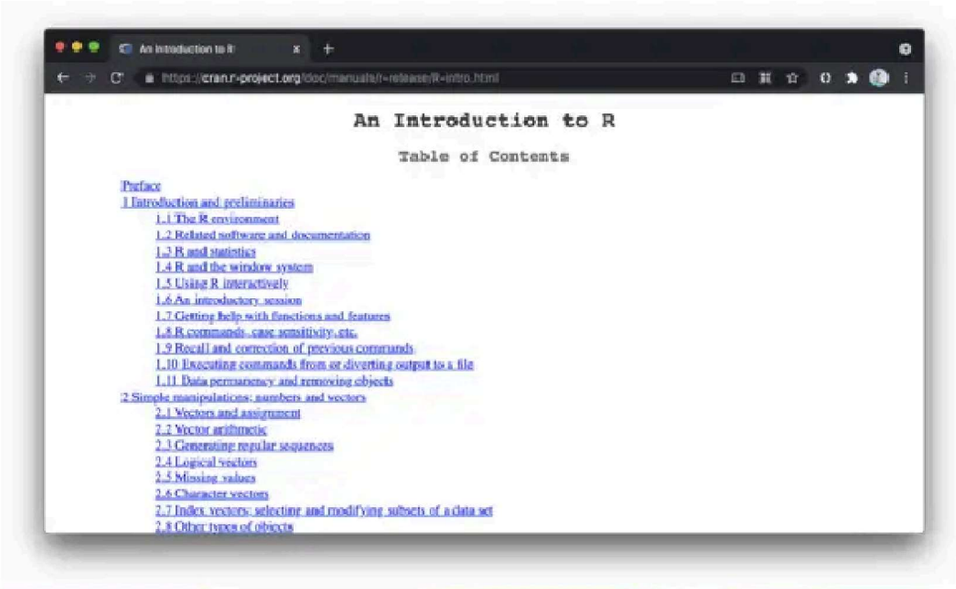
Một mạng kết nối là yêu cầu bắt buộc để một Website có thể làm việc. **Hầu hết** các Website được truy cập thông qua mạng **Internet** (Mạng toàn cầu). Tuy nhiên trong một số trường hợp Website có thể triển khai trong một **mạng nội bộ** (mạng **LAN** - **L**ocal **A**rea **N**etwork) khi máy chủ và máy khách cùng nằm trên một mạng nội bộ. Một số loại mạng phổ biến có thể sử dụng để triển ứng dụng bao gồm:

- **Mạng cục bộ (LAN-Local Area Network)** kết nối các máy tính trong phạm vi gần, thường là một mạng máy tính nội bộ trong một tổ chức.
- **Mạng diện rộng (WAN-Wide Area Network)** gồm nhiều mạng LAN kết nối để tạo ra một mạng với quy mô lớn hơn có thể là một thành phố hay một quốc gia.
- **Một ISP (Internet Service Provider)** là một đơn vị sở hữu, quản lý một mạng diện rộng (WAN), đồng thời cung cấp dịch vụ để truy cập vào mạng Internet.
- **Mạng Internet** bao gồm các mạng WAN được kết nối với nhau và trao đổi thông tin qua các điểm trao đổi Internet IXP (**Internet eXchange Point**), có khoảng 600 IXP trên thế giới.

Tuy nhiên, cũng có trường hợp Website hoạt động trên máy tính đơn khi đó máy tính đó vừa đóng vai trò là máy khách vừa là máy chủ.

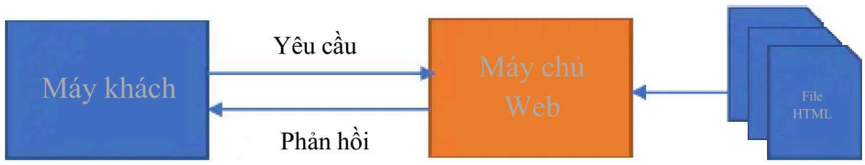
1.1.3 Phân loại Web

Ba thành phần chính để tạo lên một Website là **máy khách** - nơi có trình duyệt, **máy chủ** - nơi cung cấp dịch vụ, hai thành phần này kết nối với nhau thông qua một **hệ thống mạng.** Web được chia làm 2 loại **Web tĩnh (Static)** và **Web động (Dynamic).** Ví dụ về một trang Web tĩnh:



Hình 1-2 Một trang Web tĩnh

Một trang Web tĩnh được xử lý như sau:



Hình 1-3 Cách thức làm việc của một trang Web tĩnh

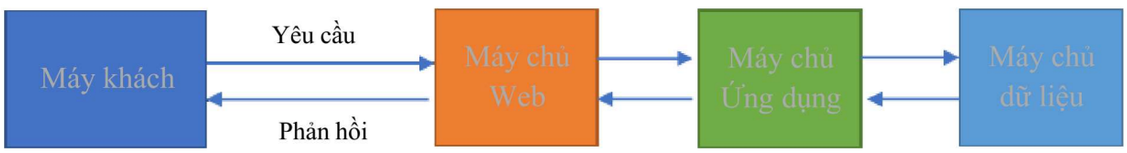
- Người dùng gửi các yêu cầu (request) bằng trình duyệt thông qua giao thức HTTP.
- Server gửi trả lại (respond) là trang tài liệu HTML.
- Trình duyệt kết xuất (render) và hiển thị nội dung trang tài liệu HTML.

Một trang Web tĩnh một trang Web mà nó chỉ thay đổi khi lập trình viên sửa đổi nội dung của nó. Những trang Web này đặt trên máy chủ, thường dưới định dạng của tập tin **HTML** (**Hypertext Markup Language** - ngôn ngữ đánh dấu siêu văn bản). HTML là một ngôn ngữ được thiết kế ra để tạo nên các trang Web. Một trang Web tĩnh là một trang HTML được lưu trên máy chủ. Khi người dùng gửi yêu cầu tới máy chủ thông qua trình duyệt, yêu cầu đó sẽ bao gồm địa chỉ đầy đủ các trang HTML (Ví dụ: https://cran.r-project.org/doc/manuals/r-release/R-intro.html). Khi máy chủ nhận được yêu cầu nó sẽ lấy trang Web tĩnh (tập tin HTML), sau đó gửi trả lại phía trình duyệt. Khi nhận được file HTML, trình duyệt sẽ tiến hành kết xuất (render) và hiển thị nội dung lên trình duyệt.

Các yêu cầu gửi đến máy chủ được gọi là các **HTTP Request**, đó là những yêu cầu được định dạng bởi **giao thức HTTP** (HyperText Transfer Protocol) hay còn gọi là giao thức truyền tải siêu văn bản. Dữ liệu máy chủ gửi trả về là **HTTP Response** trong đó chứa nội dung của trang HTML. Nếu người dùng muốn xem một trang khác ví dụ bấm vào một liên kết trên trang HTML trả về, một HTTP Request lại được tạo ra và gửi đi, quá trình như vậy sẽ được lặp đi lặp lại.

Khác với một trang Web tĩnh, Web động có những đặc điểm sau:

- Được tạo bởi một **chương trình** (program) hoặc **kịch bản** (script) trên **máy chủ Web**.
- Chương trình hoặc kịch bản được vận hành bởi một **máy chủ ứng dụng.**
- Dữ liệu có thể được cung cấp bởi một **máy chủ dữ liệu.**



Hình 1-4 Cách thức làm việc của một trang Web động

Một trang Web động được tạo ra khi người dùng yêu cầu chứ không phải có sẵn từ trước. Những thông tin được trả về cho người dùng sẽ phụ thuộc vào yêu cầu của người dùng gửi đến máy chủ Web (các HTTP Request), máy chủ Web có thể kết nối tới nhiều các **máy chủ ứng dụng** và **máy chủ dữ liệu**.

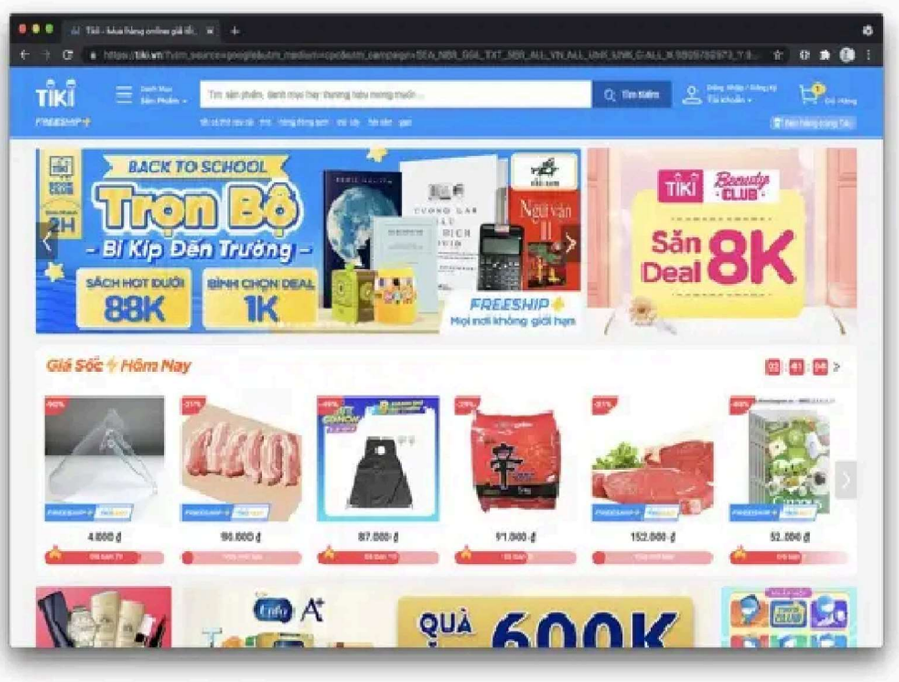
Khi nhận được yêu cầu, máy chủ Web sẽ lựa chọn **máy chủ ứng dụng** nào phù hợp để xử lý yêu cầu. Khi xử lý yêu cầu, Máy chủ ứng dụng sẽ chạy những đoạn mã lệnh chương trình hoặc kịch bản để tạo ra các trang HTML. Máy chủ ứng dụng có thể liên kết đến các các máy chủ dữ liệu để sử dụng dữ liệu tạo ra trang HTML.

Khi xử lý xong các yêu cầu, trang HTML được gửi lại cho người dùng thông qua một **HTTP Response**. Như vậy các trang HTML là không được tạo sẵn từ trước, mà chỉ được tạo ra trong quá trình xử lý yêu cầu của người dùng. Khi nhận được HTTP Response, trình duyệt sẽ đọc nội dung HTML và hiển thị cho người dùng. Cần lưu ý rằng trình duyệt hoàn toàn không quan tâm đến trang việc đây là Web tĩnh hay Web động, nhiệm vụ của trình duyệt là gửi các yêu cầu và

hiển thị trang HTML mà nó nhận được từ phía Máy chủ. Khi trang Web được hiển thị trên trình duyệt, người dùng có thể bấm vào các liên kết trên trang Web đó để gửi những yêu cầu khác tới Máy chủ. Quá trình đó lặp đi lặp lại trong quá trình người dùng sử dụng. Một website động giúp người dùng tương tác với website nhằm thực hiện một công việc. Vì vậy Website động còn được gọi là ứng dụng Web.

Các máy chủ Web phổ biến hiện nay gồm có IIS, Apache, Nginx... Các công nghệ để để xây dựng các trang Web động được gọi là các công nghệ phía máy chủ, thông dụng hiện nay gồm có JSP, ASP, PHP … Các công nghệ này sẽ đi kèm với việc tạo ra các Máy chủ ứng dụng để thực hiện nhiệm vụ tạo ra các trang HTML. Các máy chủ dữ liệu phổ biến hiện nay như MySQL, PostgreSQL, SQL Server, MongoDB...

Ví dụ về một trang Web động:



Hình 1-5 Ví dụ một trang Web động

1.1.4 Đường dẫn Web

Đường dẫn Web hay còn được gọi là **URL**:

- URL là viết tắt của **U**niform **R**esource **L**ocator.
- URL không khác gì địa chỉ của một tài nguyên duy nhất nhất định trên Web.
- Mỗi URL trỏ đến một tài nguyên duy nhất. Các tài nguyên đó có thể là trang HTML, tài liệu CSS, hình ảnh, v.v.
- URL có thể trỏ đến tài nguyên không còn tồn tại hoặc đã chuyển đi.

Ví dụ về các URL:

- https://developer.mozilla.org

- https://developer.mozilla.org/en-US/docs/Learn/
- https://developer.mozilla.org/en-US/search?q=URL

Một đường dẫn Web bao gồm các thành phần:

Hình 1-6 Các thành phần của đường dẫn Web

Scheme cho biết giao thức mà trình duyệt phải sử dụng để yêu cầu tài nguyên (giao thức là một phương thức thiết lập để trao đổi hoặc truyền dữ liệu trong mạng máy tính). Thông thường đối với các trang Web, giao thức là HTTPS hoặc HTTP.

Authority bao gồm địa chỉ và cổng. Địa chỉ xác định Webserver nào sẽ sử dụng để giải quyết các yêu cầu. Địa chỉ có thể xác định bởi tên miền như **www.example.com** hoặc địa chỉ IP như 102.39.23.201. Cổng là một số nguyên không dấu 16 bit, với 16 bit phạm vi cổng là từ 1 đến 65535 (số cổng 0 được bảo lưu và không thể được sử dụng). Với việc sử dụng cổng, Server có thể phân tán các yêu cầu đến nhiều dịch vụ hoạt động trên Server. ví dụ dịch vụ Web với giao thức **HTTP** sử dụng cổng **80**, giao thức **HTTPS** sử dụng cổng **443.**

Phần **Path** là đường dẫn đến tài nguyên trên máy chủ Web. Trong thời kỳ đầu path thường gắn liền với đường dẫn vật lý trên máy chủ (thư mục), tuy nhiên ngày này điều này không còn đúng nữa khi hầu hết các nội dung Web được tạo ra động (tạo ra bởi các kịch bản khi người dùng yêu cầu).

Phần **Parameters** là các tham số được gửi tới máy chủ Web. Các tham số này là một danh sách các cặp khóa/giá trị và thường được sử dụng như dữ liệu gửi kèm tới máy chủ Web để yêu cầu máy chủ thực hiện các công việc khác nhau với dữ liệu được gửi đến.

Anchor đi kèm với dấu # hay còn gọi là hash. Một hoạt động như một mỏ neo để đánh dấu một thành phần trên tài liệu HTML giúp người dùng dễ dàng hơn trong việc điều hướng khi xem trang Web.

1.1.5 Trình duyệt Web

Trình duyệt Web là một ứng dụng phần mềm để truy cập thông tin trên mạng Internet. Trình duyệt Web đọc các tài liệu HTML được trả về từ một máy chủ Web và hiển thị nó. Khi thiết kế, các trang Web cần được kiểm tra trên các trình duyệt khác nhau. Bảng tổng kết các trình duyệt phổ biến trên thế giới tính đến tháng 10 năm 2022 (Thống kê từ Website https://statcounter.com)

Bảng 1-1 Thống kê thị phần các trình duyệt (tháng 2 năm 2023)

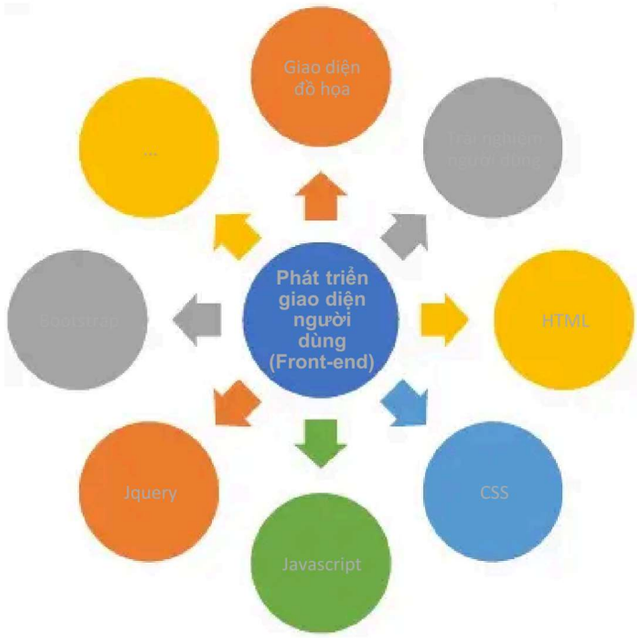
Trình duyệt	Thị phần
Google Chrome	65.74%
Apple Safari	18.84%
Microsoft Edge	4.27%

Mozilla Firefox	2.92%
Samsung Internet	2.6%
Opera	2.27
Các trình duyệt khác	0.29%

1.2 QUY TRÌNH THIẾT KẾ WEBSITE

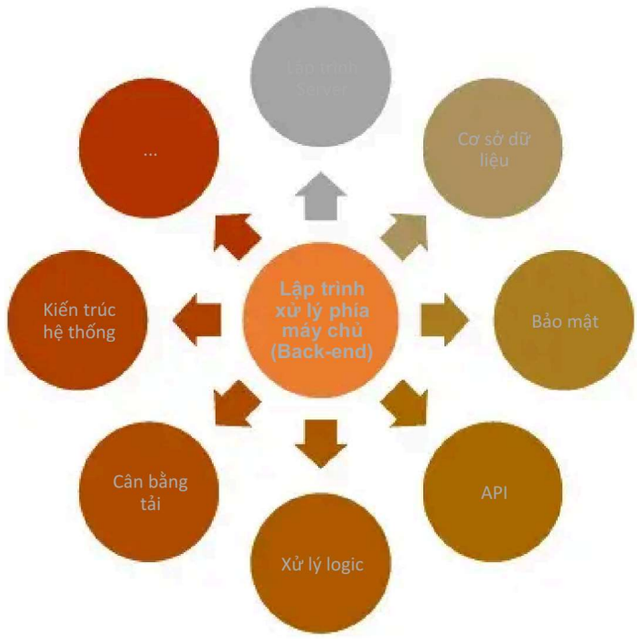
Thiết kế Website là một bước trong quá trình **phát triển Website hoàn chỉnh**. Cụ thể, để phát triển một Website hoàn chỉnh thường được chia thành 2 mảng công việc lớn là **Thiết kế giao diện người dùng (Front-end**) chính là trọng tâm của giáo trình này, và **Lập trình xử lý phía máy chủ (Back-end).**

Phát triển giao diện người dùng (Front-end): Đây là quá trình thiết kế giao diện, trải nghiệm, bằng cách sử dụng các công cụ đồ họa, các quy tắc thiết kế và các ngôn ngữ như HTML, CSS, JavaScript... nhằm xây dựng giao diện cho các trang Web để người dùng có thể xem và tương tác trực tiếp trên đó. Mục tiêu của việc chuyên môn hoá thiết kế giao diện Website là giúp cho người dùng có những trải nghiệm tốt nhất khi sử dụng một Website. Điều này rất quan trọng vì đó là những gì người dùng sẽ nhìn thấy và tương tác trực tiếp với hệ thống. Việc phát triển giao diện người dùng yêu cầu cần phải quan tâm đến các yếu tố như sau:



Hình 1-7 Phát triển giao diện người dùng (Front-end)

Lập trình xử lý phía máy chủ (Back-end): sử dụng các ngôn ngữ (PHP, ASP.net, JSP,…), cơ sở dữ liệu (MySQL, SqlServer,…), kiến thức về cấu hình/quản trị máy chủ,… nhằm giúp một Website hoạt động tốt, cung cấp đầy đủ thông tin cần thiết tới người dùng một cách nhanh chóng. Lập trình viên phát triển Back-End (Back-End Developers) giúp xử lý các nghiệp vụ phức tạp ở phía sau một cách logic, đảm bảo Website hoạt động trơn tru khi đưa vào vận hành trên thực tế.



Hình 1-8. Lập trình xử lý phía máy chủ (Back-end)

Để phát triển một Website, những công nghệ như HTML, CSS và JavaScript cung cấp cho Web chúng ta biết hình dạng của Website như thế nào và xác định cách chúng ta tương tác với thông tin. Nhưng những gì thường ẩn phía sau, vẫn là phần quan trọng trong vòng đời phát triển Website - đó là các giai đoạn thu thập thông tin từ khách hàng, lập kế hoạch chi tiết và bảo trì sau khi ra mắt. Phần này sẽ đề cập tới **quy trình 7 bước** chính của phát triển Website:

Hình 1-9 Quy trình phát triển một Website

Bước 1: Thu thập thông tin

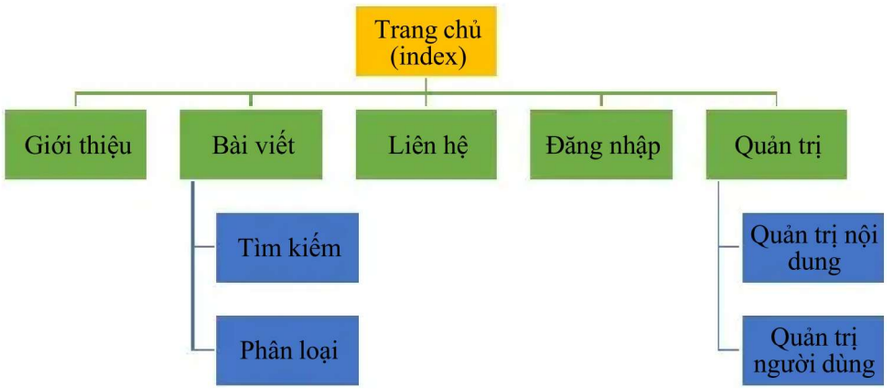
Quá trình phát triển một Website bắt đầu bằng việc **thu thập thông tin** để xây dựng Website. Các thông tin này có thể lấy từ người đặt hàng xây dựng Website, người sử dụng trong tương lai, hoặc thông qua các nghiên cứu, khảo sát. Đây là một nhiệm vụ rất quan trọng nhằm hiểu rõ mục đích trang Web, mục tiêu muốn đạt được và đối tượng người dùng sẽ sử dụng hệ thống. Các thông tin cần thu thập thường bao gồm:

- Mục đích của trang Web
- Đối tượng sử dụng

- Các tính năng chính của trang Web
- Giao diện đồ họa, công nghệ,... mong muốn sử dụng
- Ngân sách, cơ chế thanh toán/tạm ứng,…

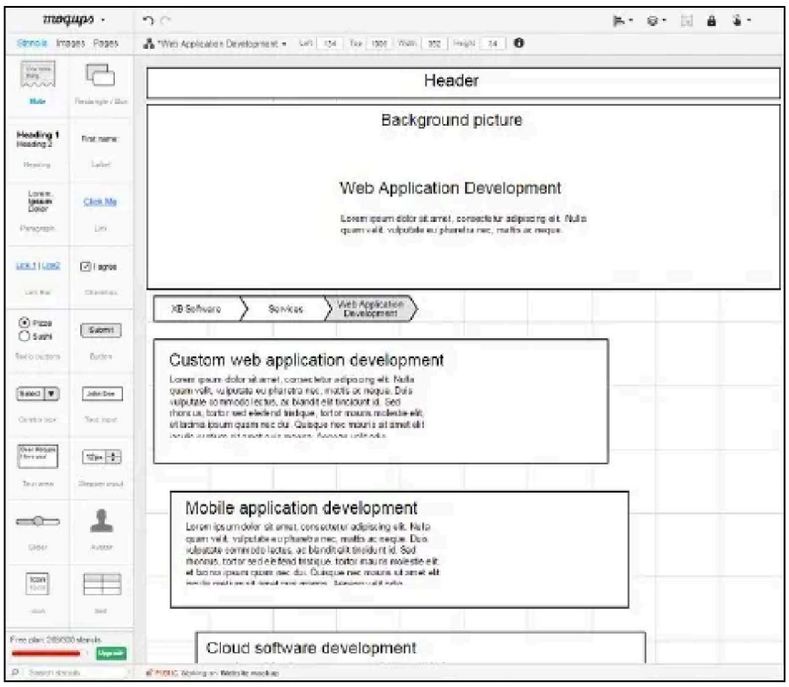
Bước 2: Lập kế hoạch

Sau khi thu thập được các thông tin từ khách hàng và đánh giá tính khả thi trong việc thực hiện. Bước tiếp theo đó là đưa ra một kế hoạch đề xuất chi tiết nhằm giúp khách hàng nắm rõ ràng về bài toán, nguồn lực, tiến độ thời gian và chi phí,… liên quan đến dự án. Ở giai đoạn này của quy trình phát triển Website, chuyên viên phân tích nghiệp vụ (BA) & khách hàng đánh giá toàn bộ trang Web sẽ trông như thế nào. Dựa trên thông tin đã được thu thập cùng nhau trong giai đoạn trước, sơ đồ trang Web (sitemap). Sơ đồ Website là vô cùng quan trọng, nó giúp cho các đối tượng như khách hàng hoặc những người tham gia phát triển, vận hành Website hình dung được các trang/chức năng chính của hệ thống. Ví dụ một sơ đồ Website như sau:



Hình 1-10 Ví dụ về sơ đồ trang Web

Sơ đồ trang Web sẽ mô tả mối quan hệ giữa các khu vực chính trên trang Web của bạn. Cách trình bày như vậy có thể giúp hiểu sản phẩm cuối cùng có thể sử dụng được như thế nào. Nó có thể cho thấy "mối quan hệ" giữa các trang khác nhau của trang Web, vì vậy có thể đánh giá mức độ dễ dàng đối với người dùng cuối trong việc tìm kiếm thông tin hoặc dịch vụ được yêu cầu nếu họ bắt đầu từ trang chính. Lý do chính đằng sau việc tạo sơ đồ trang Web là để xây dựng một trang Web thân thiện với người dùng và dễ điều hướng. Sơ đồ trang Web cho phép hiểu cấu trúc bên trong của trang Web trông như thế nào nhưng không mô tả giao diện người dùng. Vì vậy, trước khi bạn bắt đầu thiết kế hoặc lập trình, cần phải nhận được sự chấp thuận từ khách hàng rằng mọi thứ đều ổn để bạn có thể bắt đầu giai đoạn phát triển tiếp theo. Trong trường hợp này, một **"phác thảo" (Wireframe)** nên được tạo. **Phác thảo** là một đại diện trực quan của giao diện người dùng mà sẽ được tạo. Thường nó không chứa bất kỳ yếu tố thiết kế nào như màu sắc, logo, v.v. Hình bên dưới mô tả một **phác thảo** được tạo ra dựa trên công cụ Moqups.



Hình 1-11 Ví dụ một phác thảo được tạo bởi công cụ Mogups

Bước 3: Thiết kế giao diện, trải nghiệm

Sau khi thống nhất lại tất cả các khía cạnh trong lập kế hoạch triển khai hệ thống, bước tiếp theo của việc phát triển Website là bắt đầu quá trình thiết kế. Quá trình bao gồm việc thiết kế cả **giao diện người dùng (User Interface - UI)** và **trải nghiệm người dùng (User Experience UX)**. Trong đó UI dùng để mô tả giao diện người dùng, , đó là những thành phần mà người dùng tiếp sẽ tiếp xúc với như các nút bấm, biểu tượng, ô nhập ... Trong khi đó UX là trải nghiệm, cách thức người dùng tương tác với các thành phần UI được tạo ra.

Hình 1-12 Thiết kế giao diện (UI), trải nghiệm (UX) cho Website

Tất cả nội dung trực quan, như hình ảnh, ảnh và video được tạo ở bước này. Một lần nữa, tất cả thông tin thu thập được trong giai đoạn đầu tiên đều rất quan trọng. Khách hàng và đối tượng mục tiêu phải được ghi nhớ trong khi thiết kế. Bố cục của các trang Web thể hiện cấu trúc thông tin, trực quan hóa nội dung và thể hiện chức năng cơ bản. Bố cục chứa màu sắc, logo, hình ảnh và có thể cung cấp thông tin chung về sản phẩm trong tương lai. Sau đó, khách hàng có thể xem lại bố cục và gửi phản hồi cho bạn. Nếu khách hàng băn khoăn hoặc mong muốn điều chỉnh một số khía cạnh trong thiết kế của bạn, bạn nên thay đổi bố cục và gửi lại cho họ. Chu kỳ này nên được lặp lại cho đến khi khách hàng hoàn toàn hài lòng.

Bước 4: Sáng tạo nội dung

Tất cả chúng ta đều biết rằng **"content is king"** *(nội dung là vua)*. Nếu Website không thể giao tiếp với người dùng thông qua nội dung hấp dẫn và mới mẻ, thì thiết kế lạ mắt và các tính năng tiên tiến của trang Web có thể đạt được rất ít về hiệu quả *(lưu lượng truy cập, lượt đăng ký, tương tác, doanh số, lợi nhuận,...)*. Ở bước này, cần phải viết ra những điều cốt lõi mà bạn muốn truyền đạt tới người xem trang Web của mình. Viết nội dung liên quan đến việc tạo các tiêu đề hấp dẫn, chỉnh sửa văn bản, viết văn bản mới, biên dịch văn bản hiện có, v.v… Nội dung tạo có thể là trang giới thiệu, trang mô tả sản phẩm, trang chính sách, nhãn nút và mọi thứ khác có bản chất chủ yếu là văn bản. Điều này cần có thời gian và công sức. Theo thông lệ, bạn nên hướng khách hàng cam kết cung cấp nội dung trang Web. Sẽ tốt hơn khi tất cả nội dung trang Web được cung cấp trước hoặc trong quá trình lập trình Website.

Hiện nhiều bên phát triển hệ thống không đưa bước này vào vòng đời phát triển Website. Tuy nhiên, các công ty thiết kế và phát triển Website hàng đầu luôn quan tâm đến khía cạnh này.

Bước 5: Lập trình

Các công việc liên quan đến thiết kế & nội dung đã được thực hiện trong các giai đoạn trước (bước 3 & bước 4). Giai đoạn này bạn sẽ bắt đầu tạo từng trang Web. Thông thường, trang chủ được tạo trước, sau đó tất cả các trang phụ được thêm vào, theo hệ thống phân cấp trang Web đã được tạo trước đó dưới dạng sơ đồ trang Web. Tất cả các phần tử trang Web tĩnh được thiết kế trong quá trình tạo thiết kế phải được rà soát và kiểm tra. Sau đó, các tính năng đặc biệt và tính tương tác sẽ được thêm vào. Sự hiểu biết sâu sắc về mọi công nghệ phát triển trang Web mà bạn sẽ sử dụng là rất quan trọng trong giai đoạn này.

Nếu các trang Web phức tạp và được xây dựng từ đầu (không sử dụng các nền tảng, giải pháp có sẵn), giai đoạn này của vòng đời phát triển Website sẽ tiêu tốn khá nhiều thời gian. Giai đoạn này càng kéo dài, chi phí tổng thể của dự án càng lớn.

Bước 6: Kiểm thử, đánh giá

Kiểm thử có lẽ là phần thường xuyên nhất. Tất cả các liên kết nên được kiểm tra để đảm bảo rằng không có liên kết nào bị hỏng trong số chúng. Bạn nên kiểm tra mọi biểu mẫu, mọi tập lệnh, chạy phần mềm kiểm tra chính tả để tìm ra những lỗi chính tả có thể xảy ra. Sử dụng trình xác thực mã để kiểm tra xem mã của bạn có tuân theo các tiêu chuẩn Web hiện tại hay không.

Hình 1-13 Một số loại kiểm thử

Bước 7: Triển khai & Bảo trì

Sau khi bạn kiểm tra và kiểm tra lại trang Web của mình, đã đến lúc tải nó lên máy chủ (Web host). Phần mềm FTP (Giao thức truyền tệp) được sử dụng cho mục đích đó. Sau khi triển khai các tệp, bạn nên chạy một lần kiểm tra cuối cùng khác để đảm bảo rằng tất cả các tệp của bạn đã được cài đặt chính xác.

Thông thường, việc bảo trì cần dành nhiều thời gian hơn là phát triển một trang Web ban đầu. Chúng ta cần đảm bảo rằng mọi thứ hoạt động tốt, mọi người đều hài lòng và luôn sẵn sàng thay đổi/cập nhật hệ thống trong trường hợp cần thiết.

Hệ thống phản hồi được thêm vào trang Web sẽ cho phép bạn phát hiện các vấn đề có thể xảy ra mà người dùng cuối gặp phải. Nhiệm vụ ưu tiên cao nhất trong trường hợp này là khắc phục sự cố nhanh nhất có thể. Nếu không, một ngày nào đó bạn có thể thấy rằng người dùng của bạn thích sử dụng một trang Web khác hơn là chịu đựng sự bất tiện này.

Một số công việc liên quan đến giai đoạn bảo trì Website gồm:

- Lập kế hoạch thay đổi (Bước 1, Bước 2)
- Thiết kế thay đổi (Bước 3, Bước 4)
- Lập trình, kiểm thử và chạy thay đổi (Bước 5, Bước 6)

1.3 CÁC VẤN ĐỀ CẦN QUAN TÂM KHI THIẾT KẾ WEBSITE

Khi xây dựng Website, có **5 yếu tố** cần quan tâm đặc biệt bao gồm:

- Người dùng và tính khả dụng
- Khả năng tương thích của trang Web
- Khả năng tiếp cận mọi đối tượng người dùng
- Tối ưu hóa công cụ tìm kiếm
- Thiết kế Web đáp ứng

1.3.1 Người dùng và tính khả dụng

Trước khi thiết kế một trang Web, cần phải suy nghĩ xem **người dùng là ai** và họ sẽ mong đợi điều gì. Chính người dùng là người sẽ quyết định sự thành công của trang Web. Trang Web cần được thiết kế dễ sử dụng để người dùng có được cái họ cần ngay trong những lần vào đầu tiên. Có một bộ các nguyên tắc khi thiết kế Website mà mọi nhà phát triển cần phải tuân theo. Trong đó các nguyên tắc thiết kế cơ bản bao gồm:

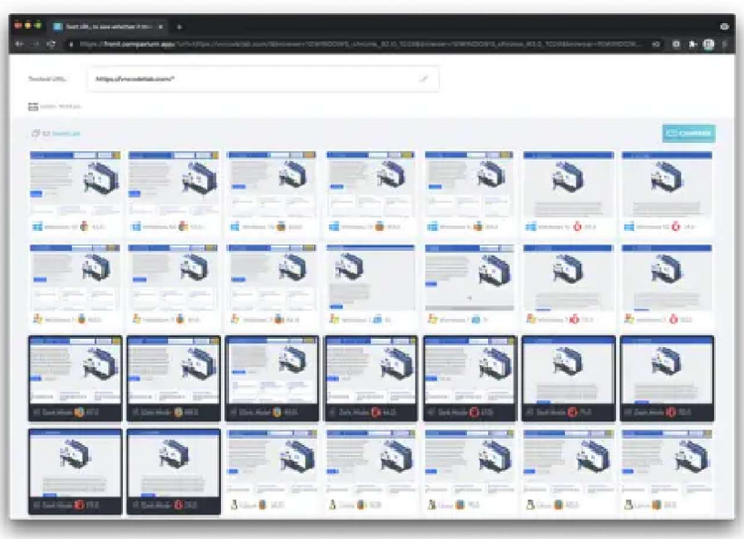
- Trình bày càng nhiều thông tin quan trọng càng tốt "trong màn hình đầu tiên" để người dùng phải cuộn ít hơn.
- Nhóm các mục liên quan và giới hạn số lượng nhóm trên mỗi trang.
- Luôn có tiêu đề xác định nội dung trang Web và cung cấp thanh điều hướng và liên kết đến các tiện ích.
- Sử dụng các quy ước điều hướng, chẳng hạn như bấm vào logo trên cùng để quay về trang chủ, bấm vào biểu tượng giỏ hàng để tới trang vật phẩm đang mua.

1.3.2 Khả năng tương thích của trang Web

Để đảm bảo Website tương thích trên nhiều trình duyệt, cần kiểm tra

- Website hoạt động trên nhiều loại trình duyệt khác nhau
- Website hoạt động trên nhiều loại hệ điều hành khác khau
- Website hoạt động trên nhiều loại thiết bị khác nhau
- Website hoạt động trên nhiều phiên bản hệ điều hành, trình duyệt khác nhau.

Có nhiều công cụ để hỗ trợ việc kiểm tra, ví dụ kiểm tra hiển thị của trang trủ trang https://vncodelab.com trên các phiên bản trình duyệt khác nhau bằng cách sử dụng https://www.comparium.app



Hình 1-14 Kiểm tra sự tương thích của trang Web trên nhiều loại trình duyệt

1.3.3 Khả năng tiếp cận mọi đối tượng người dùng

Khi thiết kế Website cần chú ý để có thể tiếp cận tới mọi đối tượng người dùng bao gồm cả những người khuyết tật:

- Những người có vấn đề về thị lực
- Những người có vấn đề về thính lực
- Những người có vấn đề về vận động
- Những người có vấn đề về trí nhớ, nhận thức
- Những người sử dụng những thiết bị đặc biệt để truy cập

Có thể thấy rằng, Việc bố trí thông tin quan trọng trên trang Web để có thể thấy ngay mà không cần phải cuộn trang có thể giúp người sử dụng có thị lực kém giảm bớt các thao tác kéo cuộn, phóng to. Việc này cũng giúp cho những người bị khuyết tật về nhận thức như những người bị mất trí nhớ tạm thời. Việc bố trí các yếu tố nội dung quan trọng trước khi cuộn trang cũng giúp cho các yếu tố nội dung được bố trí ở vị trí nhất quán. Đối với những thành phần cần xác thực nên đưa cho người dùng nhiều lựa chọn khác nhau như xác thực bằng hình ảnh, âm thanh, hoặc trả lời một số câu hỏi.Đối với những hoạt động cần các hoạt động khéo léo như thao tác trên màn hình cảm ứng hay bấm các tổ hợp phím, các thao tác giữ kéo thả cũng cần được xem xét khi thiết kế.

1.3.4 Tối ưu hóa công cụ tìm kiếm

Tối ưu hóa công cụ tìm kiếm (**SEO** - **S**earch **E**ngine **O**ptimization) ám chỉ việc thay đổi các khai báo, nội dung, cách trình bày Web để tăng thứ hạng khi tìm kiếm Web bởi các công cụ như Google, Bing và Yahoo. SEO là một yếu tố quan trọng quyết định sự thành công của một trang Web. Mặc dù các hệ thống tìm kiếm thay đổi giải thuật và cách xếp hạng các trang Web một cách khá thường xuyên, tuy nhiên có những kỹ thuật chung trong việc cải thiện, tối ưu kết quả tìm kiếm. Các thẻ title, khai báo keywords cũng như cấu trúc các thành phần trên Website quyết định rất nhiều đến SEO.

Các công cụ tìm kiếm phổ biến hiện nay

- Google
- Bing
- Yahoo

1.3.5 Thiết kế Web đáp ứng

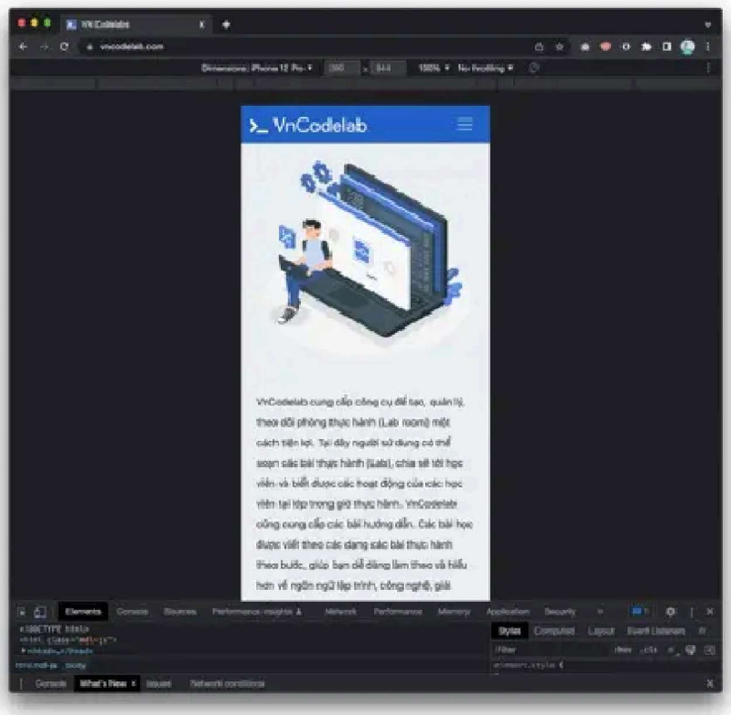
Thiết kế Web đáp ứng (**RWD** - **R**esponsive **Web D**esign) là việc thiết kế để Website có thể hiển thị dễ nhìn trên nhiều loại thiết bị khác nhau. Các loại thiết bị thông thường sẽ có các đặc tính đặc biệt là kích thước khác nhau, RWD đã trở thành một chuẩn trong thiết kế Web. nó cũng làm giảm chi phí khi thiết kế nhiều phiên bản Web khác nhau cho nhiều thiết bị khác nhau. Ngày nay người dùng thường dùng nhiều loại thiết bị khác nhau để truy cập.

Hình 1-15 Thiết kế Web đáp ứng trên nhiều loại thiết bị

Các Website cần được thiết kế đáp ứng do người dùng ngày nay sử dụng nhiều các thiết bị khác nhau để truy cập vào Web:

- Quá ½ dân số trên thế giới đã sở hữu điện thoại thông minh
- Trên ⅓ khách hàng sử dụng điện thoại thông minh
- 60% truy cập mạng là từ thiết bị di động
- 80% người dùng điện thoại thông minh truy cập các Web mua sắm trên thiết bị di động của ít nhất một lần một tháng.
- 40% người dùng sẽ rời Website và chuyển đến một trang Web khác nếu như thiết kế của Website không hiển thị hợp lý trên thiết bị của họ
- Các trang Web tìm kiếm như Google ưu tiên kết quả cho các Website có thiết kế tốt.

Thiết kế Website được thiết kế một cách dễ dàng



Hình 1-16 Kiểm tra Website hoạt động trên nhiều loại kích thước màn hình

1.4 HTML, CSS, JAVASCRIPT

HTML, **CSS**, và **JavaScript** có vai trò lớn trong thiết kế Web: **HTML** cung cấp cấu trúc, nội dung. **HTML** document có đuôi file dạng .html hoặc .htm. Một tài liệu HTML bao gồm 1 bộ tag (hay còn gọi là element). HTML quyết định một trang Web bao gồm những thành phần, nội dung gì. HTML không phải là ngôn ngữ lập trình, do đó HTML không thể tạo ra các chức năng "động" được. **CSS** kiểm soát trình bày, định dạng và bố cục. **CSS** được sử dụng để định dạng các thành phần được tạo ra bởi ngôn ngữ HTML. CSS định nghĩa ra các phong cách (style) bao gồm các thông tin về bố cục, màu sắc trang, ... và áp dụng các phong cách này cho các thành phần khác nhau. CSS có thể được khai báo trong tệp tin HTML hoặc trong tập tin CSS. **JavaScript** kiểm soát hành vi của các yếu tố khác nhau. **JavaScript** là ngôn ngữ kịch bản hoạt động ở phía máy khách, JavaScript hoạt động độc lập với máy chủ để thực hiện các kịch bản cần thiết. Tuy nhiên, JavaScript cũng được sử dụng để gửi, nhận dữ liệu từ máy chủ.

Khi người dùng gửi yêu cầu tới máy chủ, dữ liệu trả về bao gồm cả 3 loại file **HTML**, **CSS** và **JavaScript** , khi đó file **CSS** sẽ định dạng hiển thị cho các thành phần trong file **HTML**, đồng thời file **JavaScript** cũng sẽ được thực thi.

1.4.1 Tài liệu HTML

Một tài liệu HTML bao gồm các thẻ nhằm định nội dung và cấu trúc của Website:

- Tất cả các tài liệu HTML đều bắt đầu bằng khai báo **DOCTYPE**

- Toàn bộ tài liệu HTML khai báo các thẻ nằm bên trong thẻ mở **<html>** và thẻ đóng **</html>**
- Một tài liệu html cơ bản có phần **head** và **body**, trong đó head chứa các thông tin về tài liệu, và phần body chứa những nội dung sẽ hiển thị.
- Mỗi thẻ trong tài liệu tạo ra một **phần tử HTML**.
- Mỗi một **phần tử HTML** có thể có các thuộc tính để định danh và xác định nó được hiển thị hoặc có hành vi như nào.

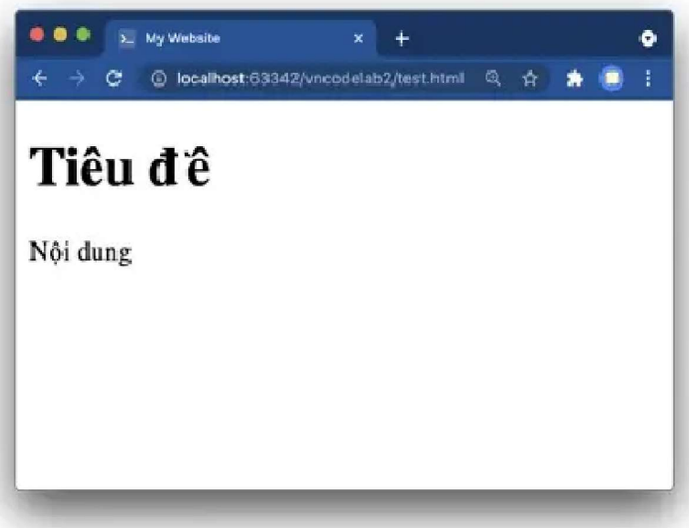
Các chuẩn HTML được phát triển bởi tổ chức quốc tế **W3C** (World Wide Web Consortium) và **WHATWG** (Web Hypertext Application Technology Working Group) xây dựng. Website của tổ chức W3C là, www.w3c.org. Trong khí đó WHATWG là một "**Nhóm làm việc về công nghệ ứng dụng siêu văn bản Web"** bao gồm các thành viên chính là các công ty Apple, Mozilla, Google và Microsoft, Website: https://www.whatwg.org. HTML được phát triển từ năm 1993 đã trải qua nhiều phiên bản:

- **HTML 1.0:** Ra đời năm 1993 và chưa bao giờ trở thành một chuẩn
- **HTML 2.0:** Phát triển và sử dụng từ năm 1995 đến 1996, còn được gọi là chuẩn RFC 1866. Bao gồm các đặc tính chính (core) như các thẻ html, head, body, p, ul, li, ...
- **HTML 3.2:** Được công bố năm 1996, Chuẩn HTML 3.2 đưa các thêm các đặc tính mới như css, script, table, applets, ...
- **HTML 4.0:** Chuẩn HTML phiên bản 4 được công bố năm 1997 hỗ trợ một cách đầy đủ hơn dữ liệu đa phương tiện, các ngôn ngữ kịch bản, và các phiên bản CSS 2, CSS 3.
- **XHTML 1.0:** Phiên bản được công bố năm 2000 đã cải tổ HTML 4 sử dụng cú pháp XML, giúp cho việc đọc, và phân tích cú pháp tài liệu HTML được dễ dàng hơn
- **XHTML 1.1:** Phiên bản mới của XHTML được công bố năm 2001 bổ sung thêm các yêu cầu chặt chẽ hơn.
- **HTML5:** được phát triển từ 2008 kế thừa cả XHTML để tạo thành một chuẩn mới. HTML 5 bổ sung thêm nhiều cú pháp và thẻ mới như <video>, <audio>, <canvas>, và tích hợp thêm đồ họa vector. HTML5 vẫn tiếp tục được phát triển và có những chuẩn mới như HTML 5.1, HTML 5.2, HTML 5.3

Ví dụ về một tài liệu HTML mẫu:

```
<!DOCTYPE html>
<html>
<head>
   <meta charset="UTF-8">
   <title>My Website</title>
</head>
<body>
   <h1>Tiêu đề</h1>
  Nội dung
</body>
</html>
```

Tài liệu HTML trên khi hiển thị ra trình duyệt:



Hình 1-17 Một tài liệu HTML đơn giản được hiển thị bởi trình duyệt

Trong tài liệu trên, khai báo **<!DOCTYPE html>** cho biết đây là một tài liệu HTML5, đôi khi người thiết kế Web có thể không đưa khai báo này vào, khi đó trình duyệt sẽ tự động xác định đây là một tài liệu HTML và phiên bản của nó. Điều này có thể dẫn đến việc hiển thị tài liệu không chính xác. Trong phần **<head>** thường khai báo các thông tin như metadata hay còn gọi là thông tin về dữ liệu. Các thông tin trong head có thể bao gồm tiêu đề, các liên kết css, thông tin miêu tả, từ khóa ... của tài liệu. Thẻ **<body>** định nghĩa phần thân hay phần nội dung của tài liệu, khi ta áp dụng các thuộc tính cho thẻ body thì thuộc tính đó sẽ mặc định áp dụng cho toàn bộ các thành phần nội dung của tài liệu.

1.4.2 Ngôn ngữ định dạng CSS

CSS được ra đời từ năm 1996, có 3 phiên bản css chính:

- **CSS 1** ra đời vào tháng 12 năm 1996 cho phép định dạng font chữ màu sắc, căn lề, trang trí viền, thiết lập khoảng cách … Hiện nay CSS1 đã ko còn được bảo trì bởi W3C
- **CSS 2** ra đời vào tháng 5 năm 1998, kế thừa nhiều định dạng từ CSS 1 đồng thời bổ sung việc thiết lập vị trí. CSS2 cũng có phiên bản CSS 2.1 trong đó thay đổi, bổ sung một số nội dung và sửa lỗi cho CSS 2
- **CSS3** là phiên bản được phát triển từ năm 1999 song song cùng với CSS 2. CSS3 bổ sung nhiều định dạng mới như bo tròn góc, đổ bóng, chuyển động, cũng như các khai báo để cấu hình bố cục như flexbox hay grid layout

Mỗi phiên bản CSS có bổ sung hoặc bỏ đi các khai báo thuộc tính khác nhau. Các phiên bản trình duyệt cũng hỗ trợ việc sử dụng các thuộc tính này:

Property	Edge	Firefox	Chrome	Safari	Opera
A					
align-content	18	28	21	9	12.1
align-items	18	20	21	9	12.1
align-self	18	20	21	9	12.1
all	79	27	37	9.1	24
animation	10	16	43	9	30
animation-delay	10	16	43	9	30
animation-direction	10	16	43	9	30
animation-duration	10	16	43	9	30
animation-fill-mode	10	16	43	9	30
animation-iteration-count	10	16	43	9	30
animation-name	10	16	43	9	30
animation-play-state	10	16	43	9	30
animation-timing-function	10	16	43	9	30
B					
backface-visibility	10	16	36	-webkit- 4	23
background	4	1	1	1	3.5
background-attachment	4	1	1	1	3.5
background-blend-mode	79	30	35	7.1	22
background-clip	9	4	4	3	10.5
background-color	4	1	1	1	3.5

Hình 1-18 Bảng hỗ trợ thuộc tính CSS của mỗi trình duyệt

Bảng đầy đủ về hỗ trợ các thuộc tính CSS của các trình duyệt có thể xem tại: https://www.w3schools.com/cssref/css3_browsersupport.asp

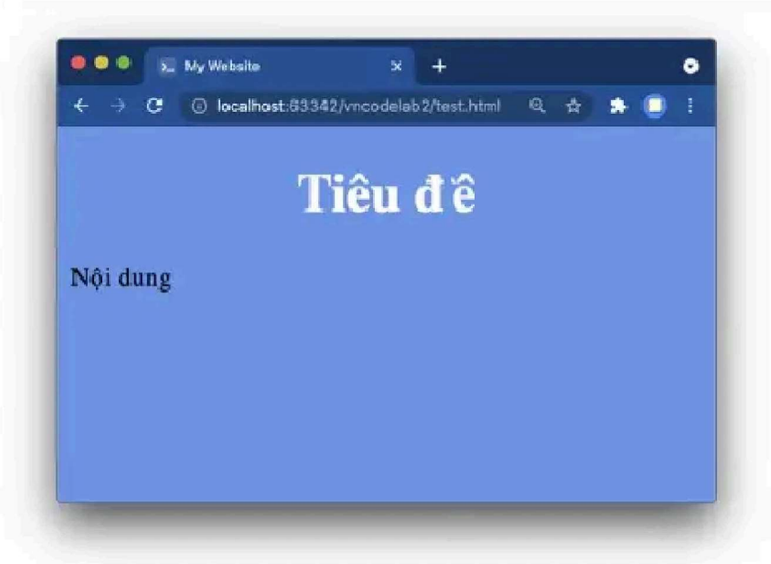
Ví dụ một tài liệu **css** được lưu trữ trong file **test.css**:

```
body {
   background-color: cornflowerblue;
}
h1 {
   color: white;
   text-align: center;
}
p {
   font-family: verdana;
   font-size: 20px;
}
```

Trong thời kỳ đầu, các tài liệu HTML định nghĩa cả cấu trúc, nội dung và định dạng. Điều này gây ra sự khó khăn khi biên tập. Ngày nay người ta thường tách riêng phần định dạng ra và lưu trong các file css. Có 3 cách để khai báo và sử dụng css (sẽ được trình bày chi tiết ở chương về CSS), tuy nhiên, cách thức phổ biến nhất để áp dụng css cho một tài liệu HTML là sử dụng file css. Khi đó tài liệu HTML sẽ sử dụng thẻ link để liên kết tới file css, thuộc tính href được sử dụng để tài liệu HTML biết sẽ phải liên kết đến file css nào. Thuộc tính rel cho biết mối quan hệ (**relationship**) giữa tài liệu HTML và tài liệu được liên kết đến là stylesheet, nghĩa là liên kết được sử dụng để định dạng phong cách cho tài liệu. Ví dụ đưa tệp định dạng **css** vào tài liệu **html**:

```
<!DOCTYPE html>
<html>
<head>
   <meta charset="UTF-8">
   <title>My Website</title>
   <link href="test.css" rel="stylesheet">
</head>
<body>
<h1>Tiêu đề</h1>
Nội dung
</body>
</html>
```

Tài liệu HTML sau khi áp dụng css:



Hình 1-19 Một tài liệu HTML được áp dụng CSS

1.4.3 JavaScript

JavaScript là ngôn ngữ lập trình được sử dụng rộng rãi ở cả phía máy khách phía máy khách (trên trình duyệt) và phía máy chủ (Nodejs). Ở phía máy khách **JavaScript** là một ngôn ngữ lập trình, được tích hợp và nhúng trong HTML giúp Website sống động hơn. JavaScript cho

phép kiểm soát các hành vi của trang Web tốt hơn so với khi chỉ sử dụng mỗi HTML. **JavaScript** và **CSS** là các thành phần không thể thiếu khi phát triển Website. Khi trang HTML trả về có cần tới các file CSS và JavaScript . Trình duyệt sẽ tự động gửi yêu cầu tới Server để tải các file này về. Các file JavaScript và CSS tải về sẽ phục vụ cho việc Kiết xuất (Render) để hiển thị trên trình duyệt.

Hình 1-20 Cách thức một file JavaScript được tải về và thực hiện

JavaScript được sử dụng để người sử dụng tương tác với Web trên trình duyệt. Khi trình duyệt tải một Web, nó sẽ tải về mã HTML, khi phân tích cú pháp mã HTML trình duyệt sẽ tạo ra Mô hình đối tượng tài liệu (**DOM** - **D**ocument **O**bject **M**odel). Trong DOM có thẻ có các chỉ thị chỉ thị CSS và JavaScript , nếu là chỉ thị CSS nó sẽ chuyển giao cho CSS Parser, nếu là chỉ thị JavaScript nó sẽ gửi đến JavaScript Engine. Tuy nhiên quá trình chạy mã JavaScript được thực hiện sau khi các chỉ thị CSS hoàn tất. Thứ tự thực hiện sẽ theo thứ tự nó tìm thấy trên các trang Web. Khi đó các hàm được gọi, các sự kiện sẽ được kích hoạt dẫn đến việc DOM được cập nhật bởi JavaScript và hiển thị tới cho trình duyệt.

1.5 CÔNG CỤ THIẾT KẾ GIAO DIỆN WEB

1.5.1 Bộ công cụ của Adobe

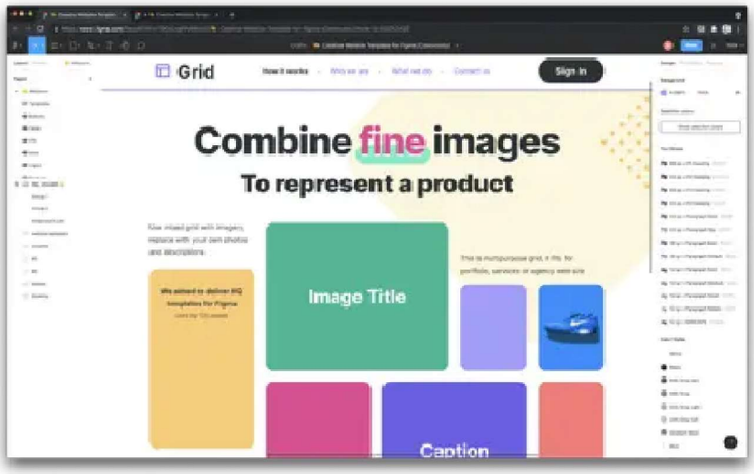
Adobe cung cấp bộ công cụ để thiết kế Web một cách trực quan bao gồm:

- **Adobe Dreamweaver (AD)** là công cụ phát triển Web, nó cũng cung cấp mỗi trường để phát triển Web, tuy nhiên nó hỗ trợ thiết kế trực quan thông qua các thao tác kéo thả, đồng thời liên kết chặt chẽ với các ứng dụng Adobe Photoshop và Adobe Illustrator.
- **Adobe Photoshop (AP)** là công cụ chỉnh sửa ảnh nổi tiếng, tuy nhiên nó cũng là công cụ quen thuộc cho các nhà thiết kế Web, Photoshop cho phép thiết kế và xuất ra file định dạng html.
- **Adobe Illustrator (AI)** cũng là một công cụ ưa chuộng để thiết kế Web, khác với AP, AI hoạt động với dữ liệu vector. Vì vậy nó phù hợp để thiết kế các trang Web với giao diện phẳng. Ngoài ra AI cũng được sử dụng nhiều để tinh chỉnh trực quan cho các đối tượng vector khi làm việc với HTML5

Hình 1-21 Biểu tượng bộ 3 công cụ của Adobe

1.5.2 Công cụ Figma

Figma.com là công cụ trực tuyến khá thông dụng để thiết kế giao diện Website. Figma lưu các thiết kế dưới dạng dữ liệu vector. Nó hỗ trợ làm việc nhóm theo thời gian thực, với việc lưu trữ nhiều phiên bản giúp việc hồi tác được dễ dàng. Ngoài ra Figma cũng có kho Plugin với nhiều Plugin hữu dụng.



Hình 1-22 Công cụ Figma

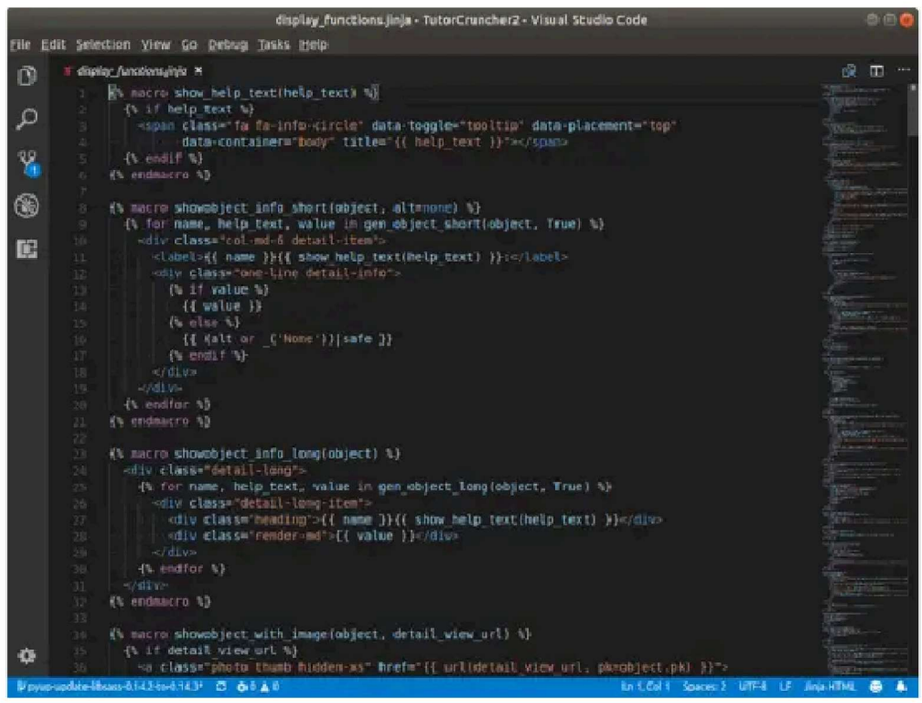
1.6 CÔNG CỤ VIẾT MÃ LỆNH THIẾT KẾ

Có rất nhiều công cụ để tạo và chỉnh sửa các file HTML, CSS và JS. Để xây dựng một Website, nhà phát triển nên có một môi trường phát triển tích hợp (**I**ntegrated **D**evelopment **E**nvironment - **IDE**). Tại đó tích hợp sẵn các công cụ biên tập, hỗ trợ lập trình, biên dịch, gỡ rối, kiểm thử, triển khai. Một IDE cũng có thể tích hợp kèm trình duyệt để hiển thị Web. Các IDE có thể miễn phí hoặc trả phí. Một số các IDE phổ biến được sử dụng nhiều trong phát triển Web bao gồm: Visual Studio Code, Sublime Text, JetBrain....

1.6.1 Công cụ Visual Studio Code

Visual Studio Code (VSC) là một công cụ được phát triển bởi Microsoft, có thể tải VSC miễn phí tại: https://code.visualstudio.com. VSC hỗ trợ xây dựng Website với nhiều ngôn ngữ lập trình và công nghệ khác nhau. VSC cũng có nhiều các Plugin hỗ trợ.

Màn hình chụp giao diện công cụ Visual Studio Code:

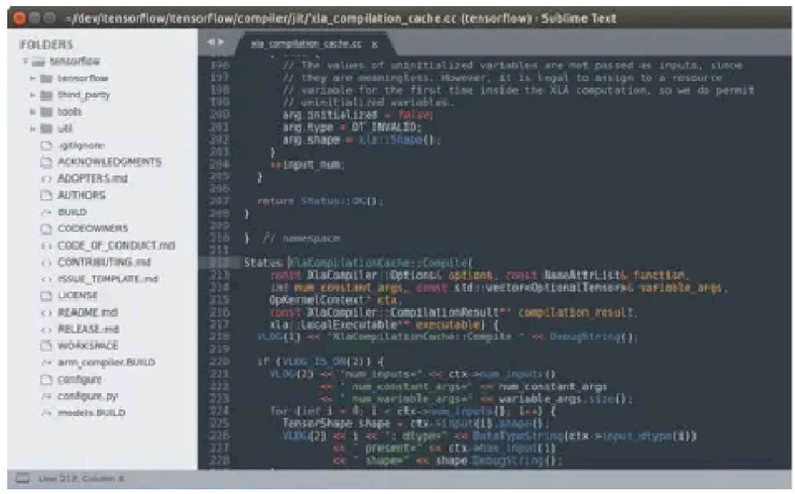


Hình 1-23 Công cụ Visual Studio Code

1.6.2 Công cụ Sublime Text

Sublime Text là một công cụ biên tập nhẹ, cài đặt và sử dụng dễ dàng. Nó hoạt động tốt trên nhiều nền tảng MAC OSX, Linux, và Windows. Sublime Text có thể được tải tại https://www.sublimetext.com/

Hình chụp giao diện công cụ Sublime Text

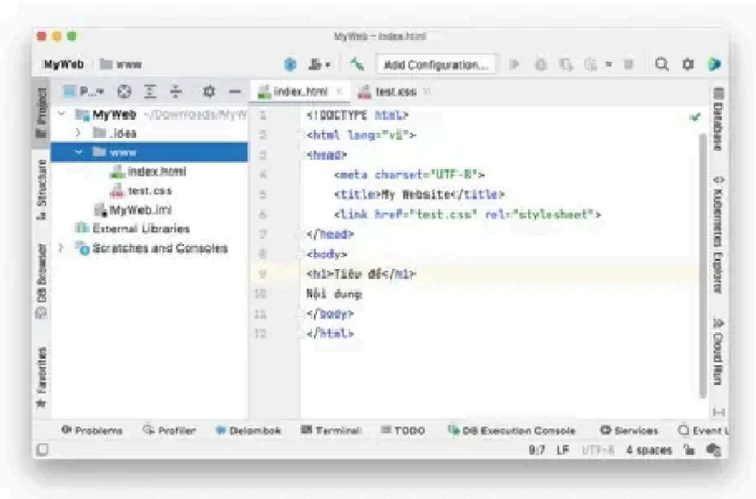


Hình 1-24 Công cụ Sublime Text

1.6.3 Bộ công cụ của JetBrains

Jetbrain là một công ty nổi tiếng với các phần mềm, công cụ lập trình và phát triển. Các công cụ của JetBrains chuyên nghiệp, dễ sử dụng, có khả năng kiểm tra tự động và sửa lỗi. JetBrains cung cấp nhiều công cụ hỗ trợ phát triển ứng dụng từ nhiều nền tảng khác nhau, và các công cụ này đều có thể được sử dụng để phát triển Web. JetBrains cung cấp việc sử dụng miễn phí cho Giáo viên, học sinh và sinh viên (những người sử hữu email tên miền edu)

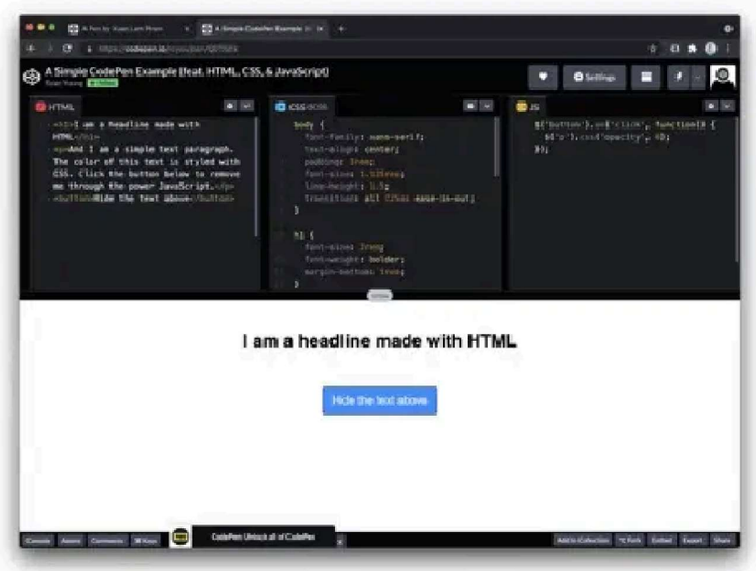
Màn hình chụp giao diện công cụ IntelliJ IDEA:



Hình 1-25 Công cụ IntelliJ IDEA của JetBrains

1.6.4 Công cụ CodePen.io

CodePen là một môi trường để học tập trình Web hoàn toàn trực tuyến, nơi có thể thử nghiệm hoặc chia sẻ những đoạn mã nguồn một cách dễ dàng.



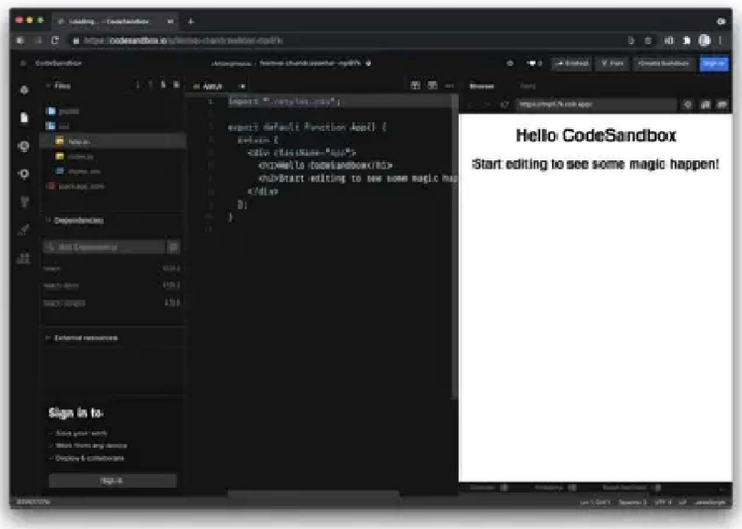
Hình 1-26 Công cụ CodePen.io

Codepen cung cấp một công cụ gọi là Pen, trong đó gồm 3 cửa sổ khác nhau cho HTML, CSS, JavaScript và một khung để xem trước kết quả. Trong quá trình thiết kế, khung xem trước được cập nhật liên tục khiến nhà phát triển dễ dàng xem được kết quả thiết kế. CodePen cũng hỗ trợ Preprocessor, add các tập lệnh từ bên ngoài, sử dụng các template sẵn có, và đặc biệt codepen có chế độ hoạt động cộng tác.

1.6.5 Công cụ CodeSandbox.io

CodeSandbox là một trình biên tập trực tuyến để phát triển nhanh Web. CodeSandbox hoạt động tương tự như các IDE được cài đặt trên máy tính. Hiện nay CodeSandbox hỗ trợ các rất nhiều các framework phổ biến, ví dụ:

- React
- Vue
- Angular
- Node HTTP Server
- ...



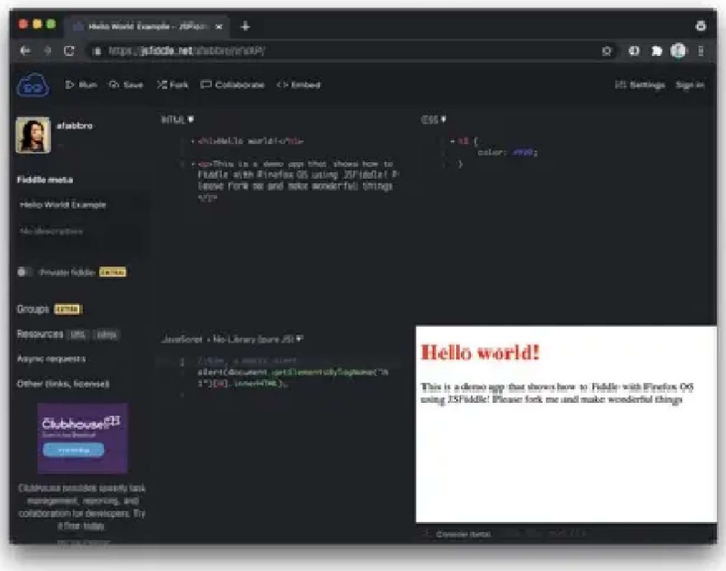
Hình 1-27 Công cụ CodeSanbox.io

Để sử dụng nhà phát triển cần tạo ra Sandbox mới từ bằng cách chọn từ các Template có sẵn. CodeSandbox.io sẽ tạo ra một Project mới với giao diện lập trình tương tự như các IDE trên máy tính. Nó bao gồm trình quản lý file, quản lý các phụ thuộc, ô soạn thảo và trình duyệt thu gọn để hiển thị kết quả. Website được gắn luôn một đường dẫn để xem trên các trình duyệt thông thường. Project đang thực hiện cũng dễ dàng chia sẻ cho các thành viên khác để cùng chỉnh sửa. Một công cụ tương tự như Codesandbox.io là stackblitz.com

1.6.6 Công cụ jsFiddle.net

Một số các đặc tính chính của **jsFiddle**:

- Hỗ trợ làm việc trực tuyến và dễ dàng chia sẻ, làm việc cộng tác
- Hỗ trợ phiên bản cho phép quay trở lại phiên bản trước đó một cách dễ dàng
- Có sẵn các thư viện thông dụng nhu jQuery, AngularJS
- Hỗ trợ nhắc lệnh tăng tốc độ lập trình, giảm thiểu sai sót.



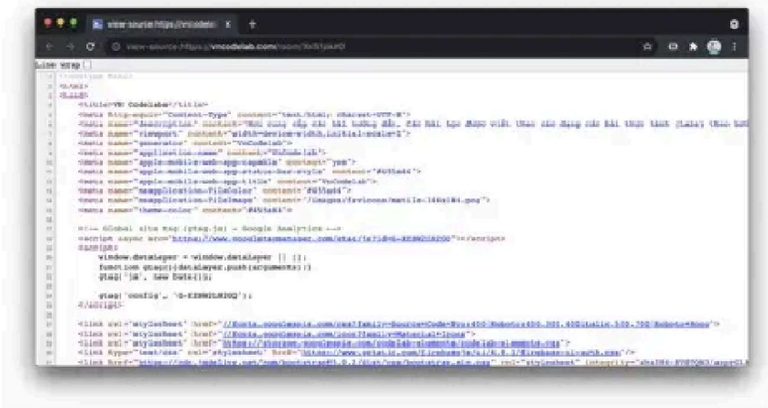
Hình 1-28 Công cụ jsFiddle

jsFiddle cũng cung cấp một màn hình gồm có các panel như HTML, CSS và JavaScript Panel. Với mỗi panel cũng có thể dễ dàng cấu hình. Như ngôn ngữ, phiên bản v.v… Bố cục của các Panel cũng có thể thay đổi như sử dụng dạng lưới 2x2 hoặc 1x3 hoặc dạng tab

1.7 CÔNG CỤ HỖ TRỢ PHÁT TRIỂN VÀ GỠ RỐI TRÊN TRÌNH DUYỆT

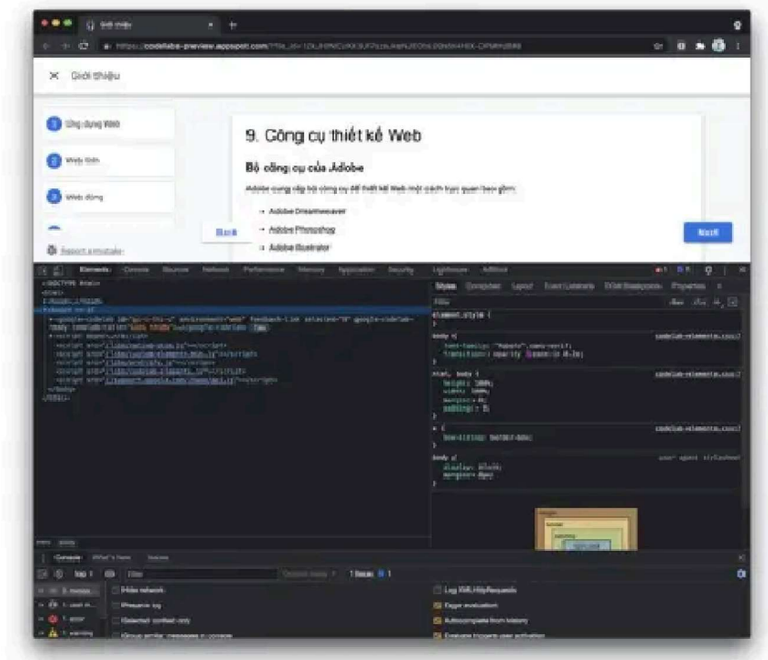
Khi Web được trả về trên Server, có thể xem mã nguồn bằng cách chọn Xem mã nguồn bằng trình duyệt:

- Xem mã nguồn HTML bằng cách bấm chuột phải và chọn View Source
- Xem mã nguồn CSS và JavaScript bằng cách bấm vào file CSS và JavaScript tương ứng



Hình 1-29 Xem mã nguồn Website trên nhiều trình duyệt khác nhau

Các công cụ phát triển (Developer Tools) cũng tích hợp sẵn trên trình duyệt để hỗ trợ phát triển và gỡ rối (Debug) các trang Web:



Hình 1-30 Công cụ Developer Tools của trình duyệt Chrome

Developer Tools được tổ chức thành các nhóm task theo định hướng trong thanh công cụ ở phía trên cùng của cửa sổ. Mỗi thanh công cụ và bảng điều khiển tương ứng cho phép bạn làm việc với một loại hình cụ thể của trang hoặc thông tin ứng dụng bao gồm cả các yếu tố DOM, resources, and sources.Các công cụ chính là có sẵn để view Developer Tools: Elements, Console, Sources, Network, Performance, Memory, Application, Security...

CÂU HỎI ÔN TẬP LÝ THUYẾT

1. HTML là viết tắt của cụm từ nào?

A. Hyper Text Markup Language
B. Home Tool Markup Language
C. Hyperlinks and Text Markup Language
D. Hyper Tool Markup Language

2. Các chuẩn HTML được phát triển bởi tổ chức nào?

A. Microsoft
B. Google
C. Mozilla
D. The World Wide Web Consortium

3. CSS là viết tắt của thuật ngữ nào?

A. Colorful Style Sheets
B. Cascading Style Sheets
C. Computer Style Sheets
D. Creative Style Sheets

4. URL là viết tắt của thuật ngữ nào?

A. Uniform Resource Locator
B. Uniform Resource Location
C. Unique Resource Locator
D. Unique Resource Location

5. Đâu KHÔNG phải là thành phần chính để tạo lên một Website?

A. Máy khách
E. Máy chủ
F. Hệ thống mạng
G. Đường dẫn Web

6. Thiết kế Web đáp ứng là gì?

A. Là việc thiết kế để Website có thể hiển thị dễ nhìn trên nhiều loại thiết bị khác nhau.
B. Là việc thiết kế để Website chỉ có thể hiển thị dễ nhìn trên 1 loại thiết bị cụ thể.
C. Là việc thiết kế để Website chỉ có thể hiển thị dễ nhìn trên 1 loại trình duyệt cụ thể.
D. Là việc thiết kế để Website có thể hiển thị dễ nhìn nhìn trên 1 loại hệ điều hành cụ thể.

7. Toàn bộ tài liệu HTML khai báo các thẻ nằm bên trong của thẻ nào?

A. Thẻ mở **<head>** và thẻ đóng **</head**

B. Thẻ mở **<body>** và thẻ đóng **</body>**

C. Thẻ mở **<html>** và thẻ đóng **</html>**

D. Thẻ mở **<main>** và thẻ đóng **</main>**

8. Khai báo <!DOCTYPE html> cho biết điều gì?

A. Đây là một tài liệu HTML2

B. Đây là một tài liệu HTML3

C. Đây là một tài liệu HTML4

D. Đây là một tài liệu HTML5

9. DOM là viết tắt của thuật ngữ nào?

A. **D**ocument **O**bject **M**ode

B. **D**ocument **O**nline **M**odel

C. **D**ocument **O**bject **M**ode

D. **D**ocument **O**nline **M**ode

10. Đâu KHÔNG phải là điểm khác biệt của một trang Web động so với một trang một trang Web tĩnh?

A. Được tạo bởi một Chương trình (Program) hoặc Kịch bản (Script) trên Máy chủ Web.

B. Chương trình hoặc kịch bản được vận hành bởi một Máy chủ ứng dụng.

C. Dữ liệu có thể được cung cấp bởi một Máy chủ dữ liệu.

D. Được tạo bởi một tài liệu HTML.

BÀI TẬP TỰ THỰC HÀNH

Xem mã nguồn của trang Web: https://minh-blogs.web.app và cho biết:

- Trang Web sử dụng bao nhiêu file css?
- Trang Web sử dụng bao nhiêu file js?

TÀI LIỆU THAM KHẢO

[1] Anne Boehm, Zak Ruvalcaba (2018) Murach's HTML5 and CSS3, Murack

[2] Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics (2018), Jennifer Robbins, O'Reilly Media

CHƯƠNG 2: NGÔN NGỮ ĐÁNH DẤU SIÊU VĂN BẢN HTML

Chương này sẽ giới thiệu về ngôn ngữ đánh dấu siêu văn bản (HTML), các phần tử cơ bản của ngôn ngữ HTML. Bên cạnh đó, người học sẽ được học cách sử dụng các phần tử HTML cơ bản để xây dựng nội dung và cấu trúc của một trang Web. Chương này cũng đề cập đến các quy ước về viết mã HTML, cách kiểm tra mã HTML để đảm bảo tài liệu HTML được viết ra là chuẩn, ngữ nghĩa, và tối ưu cho các cỗ máy tìm kiếm.

2.1 TÀI LIỆU HTML

2.1.1 Giới thiệu về ngôn ngữ HTML

HTML là ngôn ngữ đánh dấu để tạo ra các trang Web. HTML là viết tắt của **Hyper Text Markup Language**. HTML mô tả các thành phần và nội dung của các trang Web. Các phần tử HTML được biểu diễn bằng các thẻ. Trình duyệt không hiển thị các thẻ HTML nhưng sử dụng chúng để hiển thị nội dung của tài liệu HTML.

Ví dụ về một tài liệu HTML đơn giản:

```
<!DOCTYPE html>
<html>
<head>
   <title>Tiêu đề tài liệu html</title>
</head>
<body>
<h1>Tiêu đề bài viết</h1>
<p>Nội dung bài viết</p>
<img src="photo.jpg"/>
</body>
</html>
```

Trong tài liệu HTML trên:

- Khai báo **<!DOCTYPE html>** được sử dụng để đánh dấu đây là một tài liệu HTML
- Phần tử **<html>** khai báo phần tử gốc
- Phần tử **<head>** chứa các thông tin mô tả tài liệu html
- Phần tử **<title>** khai báo tiêu đề của tài liệu html
- Phần tử **<body>** chứa nội dung của tài liệu html, là phần sẽ hiển thị cho người dùng
- Phần tử **<h1>** khai báo một tiêu đề (heading)
- Phần tử **<p>** định nghĩa một đoạn văn (paragraph)
- Phần tử **<img>** tạo ra một phần tử ảnh (image)

Chú ý, khai báo **<!DOCTYPE>** chỉ xuất hiện một lần ở đầu trang (trước bất kỳ phần tử HTML nào) nó giúp khai báo loại tài liệu. Cụ thể trong trường hợp này là khai báo đây là một tài liệu HTML5**.** Ngoài khai báo này, toàn bộ các phần tử khác đều được đặt trong phần tử <**html**>.

2.1.2 Cấu trúc trang HTML

Hình 2-1 Cấu trúc một tài liệu HTML

Chỉ nội dung bên trong phần **<body>** mới được hiển thị trong trình duyệt. Phần tử **<head>** thường được sử dụng để chứa các thẻ **meta**, **title**, và **link**. Thẻ **<meta>** khai báo thông tin dữ liệu về tài liệu HTML (Siêu dữ liệu). Thẻ **<title>** chứa thông tin về tiêu đề của tài liệu trong khi thẻ **<link>** khai báo các liên kết của tài liệu tới các tài tài nguyên khác như css, hoặc ảnh faviocn. Phần tử **<head>** được đặt trong thẻ **<html>** và và trước thẻ **<body>**

2.1.3 Chú thích trong HTML

Có thể thêm chú thích vào nguồn HTML của mình bằng cách sử dụng phần tử <!--...-->. Ví dụ sau:

```
<!-- This is a comment -->
<p>This is a paragraph.</p>
<!-- Remember to add more information here -->
```

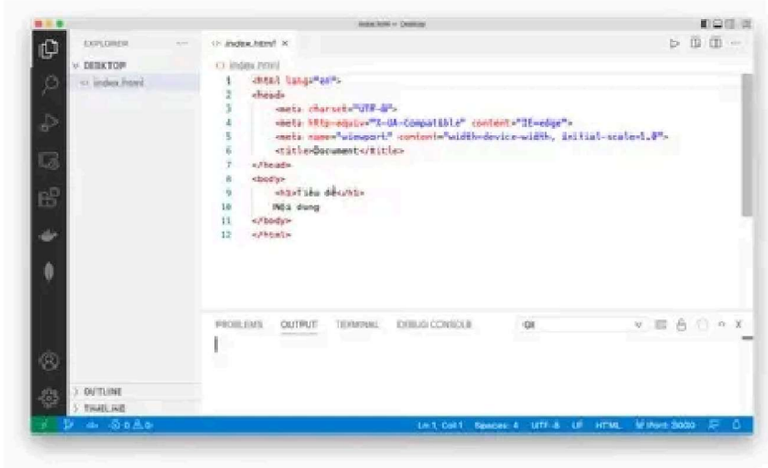
Lưu ý:

- Có một dấu chấm than (!) trong thẻ mở, nhưng không có trong thẻ đóng.
- Chú thích có có thể được viết trên nhiều dòng
- Comment không được trình duyệt hiển thị, nhưng chúng có thể giúp ghi lại, vô hiệu hóa mã nguồn HTML

2.1.4 Tạo tài liệu HTML

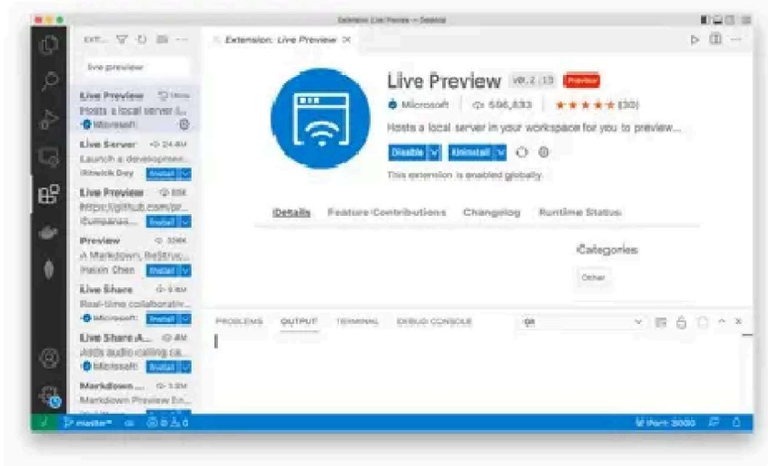
Tài liệu HTML được tạo ra bởi các kịch bản hoặc chương trình phía máy chủ (Web động) hoặc được người thiết kế tạo sẵn và lưu ở máy chủ (Web tĩnh). Các tài liệu html có phần mở rộng là *.html, được tạo ra bởi các IDE hoặc Text Editor. Các công cụ này hỗ trợ nhiều thao tác khi soạn file html, ví dụ để tạo ra một tài liệu HTML5 nhanh, có thể gõ doc rồi bấm tab trên Visual Studio Code:

Ví dụ:



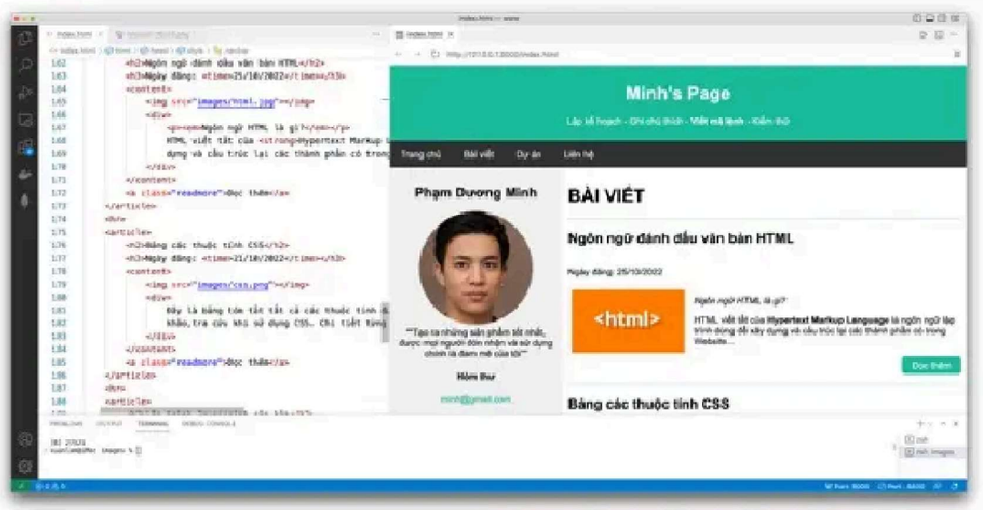
Hình 2-2 Tạo và biên tập file HTML treenVisual Studio Code

Để xem tài liệu HTML có thể dùng trình duyệt để mở file **html**. Ngoài ra, với các công cụ như Visual Studio Code, có thể bổ sung thêm các Extension để việc chạy các file HTML được dễ dàng hơn. Ví dụ công cụ Live Preview để tạo nhanh một máy chủ để xem trước một file HTML:



Hình 2-3 Cài đặt chức năng mở rộng "Live Preview" trên Visual Studio Code

Sử dụng các công cụ Live Preview dễ dàng có thể biết được tài liệu HTML được hiển thị như nào trên trình duyệt:

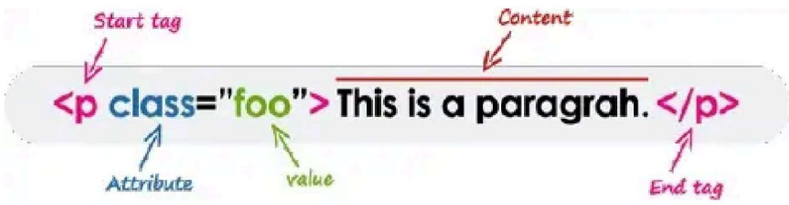


Hình 2-4 Xem trước nội dung Website bằng "Live Preview"

2.2 PHẦN TỬ HTML

2.2.1 Phần tử HTML là gì

Phần tử HTML là một loại thành phần của tài liệu HTML. Mỗi phần tử HTML được tạo từ các **thẻ HTML** bao gồm thẻ mở, thẻ đóng và các thuộc tính kèm theo. Các phần tử cũng có thể có nội dung, bao gồm các phần tử và văn bản khác.



Hình 2-5 Phần tử HTML

2.2.2 Thẻ HTML

Một phần tử HTML (HTML element) thường bao gồm **thẻ mở** và **thẻ đóng** gọi là các **thẻ HTML**, với nội dung được chèn vào giữa: <tên thẻ> Nội dung ... </ tên thẻ>. **Thẻ đóng** được viết giống như **thẻ mở,** nhưng có dấu gạch chéo được chèn trước tên thẻ. Phần tử HTML là tất cả những thứ từ thẻ mở đến thẻ đóng. Ví dụ:

Bảng 2-1 Một số thẻ mở và thẻ đóng

Thẻ mở	Thẻ đóng
<h1>	</ h1>
<p>	</ p>
<img>	Không cần
 	Không cần

HTML5 có một số phần tử chỉ cần khai báo với thẻ mở mà không cần thẻ đóng. Các phần tử này cũng không có phần nội dung và trong một số tài liệu gọi là phần tử rỗng. Các phần tử rỗng không có thẻ kết thúc, chẳng hạn như phần tử
 (cho biết dấu ngắt dòng). Các phần tử HTML có thể được lồng vào nhau (các phần tử có thể chứa các phần tử). HTML từ phiên bản 5 không yêu cầu phải đóng các phần tử rỗng. Nhưng nếu muốn xác nhận chặt chẽ hơn, hoặc nếu cần làm cho tài liệu của có thể đọc được bằng các trình phân tích cú pháp XML, cần phải đóng tất cả các phần tử HTML đúng cách.

Tất cả các tài liệu HTML bao gồm các phần tử HTML lồng nhau. Ví dụ này chứa năm phần tử HTML:

```
<!DOCTYPE html>
<html>
<body>
  <h1>Tiêu đề bài viết</h1>
  <p>Đoạn 1 bài viết</p>
  <p>Đoạn 2 bài viết</p>
</body>
</html>
```

Trong ví dụ trên, phần tử **<html>** định nghĩa toàn bộ tài liệu HTML. Nó bao gồm thẻ mở **<html>** và thẻ đóng **</html>**. Phần tử **body** xác định phần thân của tài liệu. Nó có thẻ mở **<body>** và thẻ đóng **</body>**. Nội dung của phần tử **body** là ba phần tử HTML khác (1 phần tử **h1** và 2 phần tử **p**). Phần tử **h1** định nghĩa một tiêu đề. Nó có thẻ mở **<h1>** và thẻ đóng **</h1>**. Nội dung phần tử là: "Tiêu đề bài viết". Hai phần tử **p** định nghĩa hai đoạn văn. Chúng có thẻ mở là ** ** và thẻ đóng ** **, nội dung của hai phần tử là lần lượt là "Đoạn 1 bài viết" và "Đoạn 2 bài viết".

Cần chú ý không được quên thẻ đóng mặc dù trong hầu hết trường hợp các phần tử HTML vẫn hiển thị đúng cho dù quên sử dụng thẻ đóng.

2.2.3 Thuộc tính HTML

Các thuộc tính cung cấp thông tin khai báo bổ sung cho phần tử. Các loại phần tử HTML có thể có các loại thuộc tính khác nhau. Các thuộc tính bắt buộc phải đặt trong thẻ mở. Thuộc tính thường đặt trong tên hoặc các cặp giá trị như: tên = "giá trị". Ví dụ, các liên kết HTML được định nghĩa với thẻ **<a>**. Địa chỉ liên kết được khai báo trong thuộc tính **href,** ví dụ:

```
<a href="https://www.neu.edu.vn">Liên kết đến trang www.neu.edu.vn</a>
```

Các hình ảnh được định nghĩa bằng thẻ **<img>**, ví dụ:

```
<img src="logo.jpg" width="500" height="600" alt="Logo của công ty">
```

Trong ví dụ trên, thẻ **<img>** tạo lên ảnh, tên tệp của nguồn hình ảnh được chỉ định thông qua thuộc tính **src**, kích thước hình ảnh được chỉ định bằng thuộc tính **width** và **height**. Trong ví dụ trên width = "500" có nghĩa là rộng 500 pixel và height = "600" có nghĩa là cao 600 pixel,

Thuộc tính **alt** chỉ định một văn bản thay thế được sử dụng, khi một hình ảnh không thể được hiển thị. Giá trị của thuộc tính có thể được đọc bởi trình đọc màn hình. Bằng cách này, ai đó "đang nghe" trang Web, ví dụ: một người khiếm thị, có thể "nghe" phần tử đó.

Chú ý các thuộc tính không bắt buộc đặt trong dấu ngoặc kép trừ khi thuộc tính đó có khoảng trống (dấu cách) bên trong, như trong ví dụ trên, thẻ img có thể được khai báo như sau:

```
<img src=flower.jpg width=500 height=600 alt="Bông hoa">
```

Thuộc tính “style” được sử dụng để xác định kiểu dáng của một phần tử, như màu, phông chữ, kích thước, v.v.

```
Ví dụ <p style="color:red">Đây là một đoạn </p>
```

Thuộc tính có thể có kiểu là boolean tức là chỉ nhận một trong 2 giá trị (đúng - sai, bật - tắt). Để kích hoạt thuộc tính đó là bật thì chỉ cần khai báo tên thuộc tính đó. Ví dụ một checkbox có thuộc tính **checked** thể hiện ô chọn đó sẽ được chọn sẵn (checked):

```
<input type="checkbox" name="mailList" checked>
```

Một số lưu ý khi khai báo các thuộc tính: HTML không yêu cầu tên thuộc tính phải viết chữ thường. Tức là: thuộc tính “title” có thể được viết bằng chữ hoa hoặc chữ thường như “title” hoặc “TITLE”. Tuy nhiên hầu hết sẽ được viết chữ thường để đảm bảo tính nhất quán. Các dấu nháy kép xung quanh các giá trị thuộc tính là phổ biến nhất trong HTML, nhưng cũng có thể sử dụng các dấu nháy đơn. Trong một số trường hợp, khi bản thân giá trị thuộc tính chứa dấu ngoặc kép, cần sử dụng dấu nháy đơn. Ngôn ngữ của tài liệu có thể được khai báo trong thẻ <html>:

```
<!DOCTYPE html>
<html lang="en-US">
<body>

<p title="I'm a tooltip">
This is a paragraph.
</p>

</body>
</html>
```

Ngôn ngữ được khai báo với thuộc tính “lang”. Khai báo một ngôn ngữ là quan trọng đối với các ứng dụng hỗ trợ (trình đọc màn hình) và các công cụ tìm kiếm. Ví dụ, hai chữ cái đầu tiên trong thuộc tính “lang” chỉ định ngôn ngữ (en). Nếu đó là tiếng địa phương/thổ ngữ thì sử dụng thêm hai chữ cái nữa (US). Cũng trong ví dụ trên, thuộc tính “title” được thêm vào phần tử <p>. Giá trị của thuộc tính này sẽ được hiển thị dưới dạng chú thích khi di chuột qua

2.3 QUY ƯỚC KHI VIẾT MÃ HTML

2.3.1 Một số quy ước quan trọng

Khi viết mã HTML, có một số quy ước và quy tắc cơ bản nên tuân thủ để đảm bảo mã được hiển thị chính xác trên các trình duyệt web. Dưới đây là một số quy ước khi viết mã HTML:

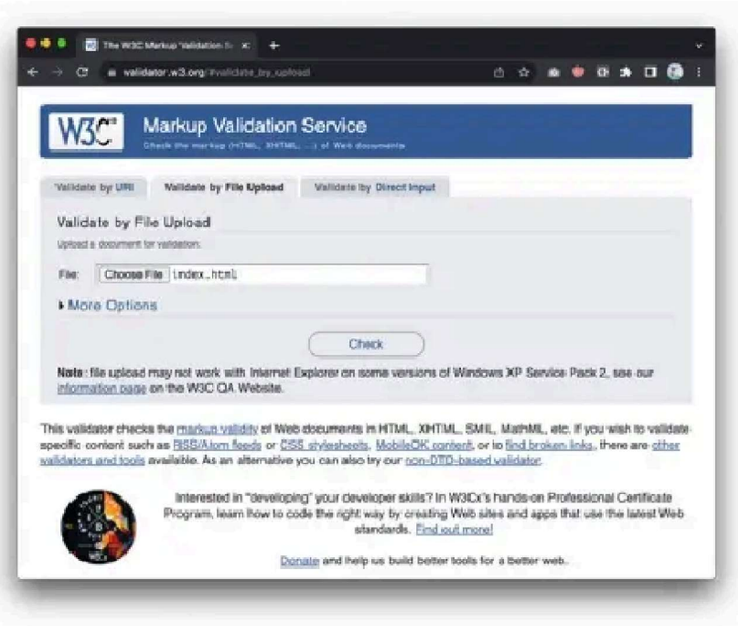
- Sử dụng tên thẻ bằng chữ thường
- Nhất quán một phong cách viết mã lệnh
- Sử dụng cú pháp gần với XHTML
- Luôn khai báo loại tài liệu bằng thẻ DOCTYPE trong dòng đầu tiên trong tài liệu
- Đóng tất cả các thành phần HTML
- Đóng các thành phần HTML trống
- Thuộc tính hình ảnh luôn khai báo đầy đủ thuộc tính alt
- Luôn luôn xác định chiều rộng và chiều cao hình ảnh
- Luôn để không gian là 1 dấu cách trước và sau dấu =
- Tránh các dòng mã dài hơn 80 ký tự
- Không thêm dòng trống mà không có lý do
- Thêm các dòng trống để tách các khối mã lớn
- Không bỏ qua các thẻ cơ bản như html, body, head, title, meta
- Sử dụng chữ thường để đặt tên tệp tin.
- Các tệp HTML phải có phần mở rộng .html hoặc .htm
- Các tệp CSS nên có phần mở rộng .css
- Các tệp JavaScript nên có phần mở rộng .js

2.3.2 Kiểm tra mã HTML

Kiểm tra mã HTML là quan trọng bởi vì nó giúp đảm bảo rằng trang web được hiển thị đúng cách trên tất cả các trình duyệt web và các thiết bị khác nhau. Việc kiểm tra mã HTML sẽ giúp tìm ra các lỗi cú pháp, thiếu thẻ đóng hoặc sai định dạng, các lỗi này có thể gây ra những vấn đề hiển thị và làm cho trang web trông không chuyên nghiệp hoặc không thể hiển thị đúng cách trên các trình duyệt khác nhau. Ngoài ra, việc kiểm tra mã HTML còn giúp xác định các vấn đề bảo mật có thể gặp phải trên trang web. Kiểm tra mã HTML cũng có thể giúp xác định các lỗi chậm khi tải trang web, giúp cải thiện tốc độ tải trang web của mình.

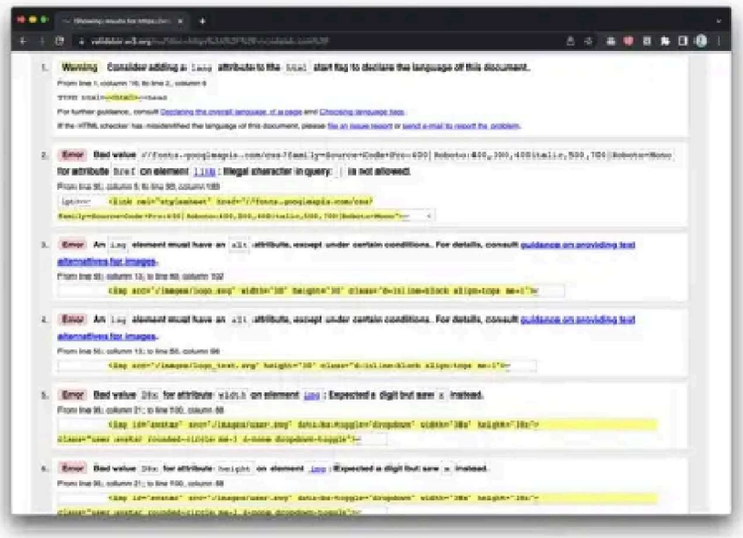
Để kiểm tra mã các lỗi khi viết mã HTML có thể sử dụng trang https://validator.w3.org. Công cụ này có thể kiểm tra:

- Một tài liệu HTML trên Server
- Một File tài liệu HTML
- Một đoạn mã HTML



Hình 2-6 Công cụ kiểm tra mã HTML

Ví dụ kết quả khi kiểm tra trang www.vncodelab.com



Hình 2-7 Các lỗi được phát hiện ra bởi công cụ Validator

2.4 PHẦN ĐẦU (HEAD) CỦA TÀI LIỆU

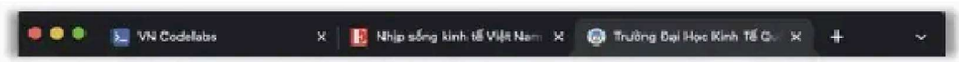
Một tài liệu liệu HTML có 2 phần chính là phần đầu (head) và phần thân (body), trong phần đầu sẽ chứa các thông tin về trang Web, trong khi phần thân sẽ chứa nội dung. Phần đầu được đặt trong phần tử **<head>** thường chứa các thông tin quan trọng sau:

- Tiêu đề của Trang Web

- Ảnh biểu tượng (xuất hiện bên cạnh tiêu đề)
- Siêu dữ liệu (metadata)
- Thông tin đường dẫn cơ sở

2.4.1 Tiêu đề

Mỗi một tài liệu HTML sẽ có một tiêu đề. Tiêu đề sẽ được trình duyệt hiển thị trên các tab của trình duyệt. Ví dụ:



Hình 2-8 Tiêu đề sẽ được xuất hiện trên các Tab của trình duyệt

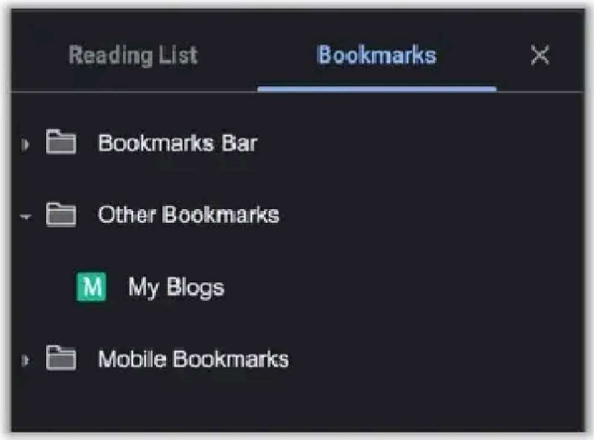
Tiêu đề được tạo ra bằng thẻ **title**, ví dụ:

```
<head>
   <title>My Shop</title>
</head>
```

Tiêu đề thông thường là tên của Website, tên của tổ chức, tuy nhiên thường thấy nhất là nó được sử dụng cho tên của bài viết. Tiêu đề có ý nghĩa rất lớn cho các cỗ máy tìm kiếm (Search engine), và sẽ xuất hiện trong kết quả tìm kiếm.

2.4.2 Ảnh biểu tượng

Biểu tượng của Website hay còn được gọi là favicon là một hình biểu tượng nhỏ được hiển thị ở góc trên cùng bên trái cửa sổ hay tab của trình duyệt hiển thị trang Web. Người dùng sẽ nhìn thấy các biểu tượng này như những hình đại diện cho chính Website. Ảnh biểu tượng được thiết kế với một kích thước chuẩn 16 x 16 pixels. Nó là một dạng rút gọn của logo, được sử dụng với mục đích nhằm giúp người dùng phân biệt được các thương hiệu một cách dễ dàng.

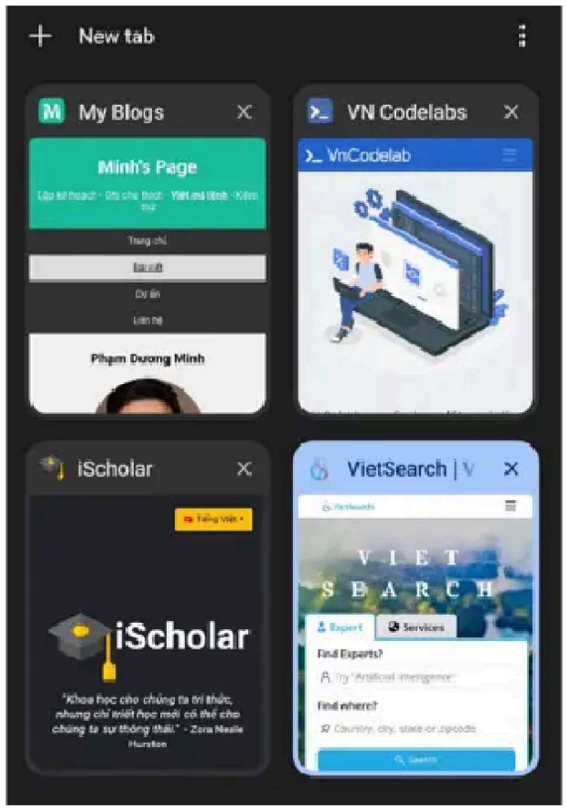


Hình 2-9 Favicon xuất hiện ở cửa sổ bookmark

Favicon được khai báo trong phần Head với thẻ <**link**>

```
<head>
  <title>My Page Title</title>
  <link rel="icon" type="image/x-icon" href="/images/favicon.ico">
</head>
```

Các ảnh icon có thể được tạo bởi các công cụ như https://favicon.io. Trong đó có hướng dẫn cấu hình để favicon này hiển thị cả trên các thiết bị di động như cửa sổ tab của điện thoại Android hoặc điện thoại iOS (Đọc phần thực hành để biết cách tạo favicon).



Hình 2-10 Các favicon cạnh tiêu đề trên cửa sổ Tab của trình duyệt trên Android

2.4.3 Các thẻ meta

Các thẻ **meta** được khai báo trong phần đầu của tài liệu để mô tả thông tin về tài liệu này, ví dụ như tác giả, từ khóa, khung nhìn.... Ví dụ khai báo thẻ meta như sau:

```
<head>
<meta charset="UTF-8">
<meta name="description" content="Hướng dẫn thiết kế Website">
<meta name="keywords" content="HTML, CSS, JavaScript ">
<meta name="author" content="Dương Minh">
<meta name="viewport" content="width=device-width, user-scalable=no, initial-
scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
</head>
```

Trong khai báo trên thẻ meta xác định charset (tập ký tự mã hóa tài liệu HTML) là UTF-8. Đây là bộ mã hóa ký tự Unicode rất phổ biến tương thích ngược với cả bộ mã ASCII. Thẻ meta này

cũng mô tả tài liệu HTML này liên quan đến Hướng dẫn thiết kế Website, và bộ từ khóa để tìm kiếm tài liệu này là HTML, CSS và Javascript. Thông tin về tác giả của tài liệu (author) và khung nhìn (viewport) tài liệu cũng được khai báo. Các thông tin này giúp cho các cỗ máy tìm kiếm (Search engine) có thể hiểu và tìm được tài liệu, ngoài ra cũng giúp trình duyệt có thêm thông tin về tài liệu để hiển thị tài liệu một cách chính xác hơn.

2.4.4 Đường dẫn cơ sở

Trong một tài liệu html sẽ có liên kết tới rất nhiều nguồn bên ngoài thông qua các đường dẫn. Ví dụ một ảnh, hoặc một liên kết. Để khai báo đường dẫn cơ sở cho các liên kết này có thể dùng phần tử <**base**>. Ví dụ các phần tử ảnh và phần tử a liên kết đến Website.com sẽ được khai báo như sau:

```
<img src="https://minh-blogs.Web.app/images/logo.png" width="24" height="39">
<a href="https://minh-blogs.Web.app/index.html">Vào trang chủ</a>
```

Có thể được viết lại thành:

```
<head>
  <base href=" https://minh-blogs.Web.app/" target="_blank">
</head>
<body>
<img src="images/logo.png" width="24" height="39">
<a href="index.html">Vào trang chủ</a>
</body>
```

2.5 PHẦN TỬ KHỐI VÀ PHẦN TỬ NỘI TUYẾN

Tất cả các phần tử trong HTML có thể được chia thành hai loại: các **phần tử khối (block)** và các **phần tử nội tuyến (inline).**

2.5.1 Phần tử khối

Phần tử khối (block) thường bắt đầu và kết thúc với một dòng mới khi hiển thị trong trình duyệt. Ví dụ cho các phần tử này là **<p>, <h1>, <h2>, <h3>, <h4>, <h5>, <h6>, <ul>, <ol>, <dl>, <pre>, <hr />, <blockquote>,** và **<address>**. Tất cả các phần tử này đều bắt đầu bởi một dòng mới, và chiếm luôn toàn bộ độ rộng của trang Web, điều này cũng đồng nghĩa với việc các phần tử đi sau phần tử này cũng sẽ phải xuất hiện trên dòng mới.

Thẻ <**div**> là một thẻ tiêu biểu được sử dụng để tạo ra một phần tử khối phi ngữ nghĩa (chúng ta sẽ tìm hiểu về phần tử ngữ nghĩa ở phần sau). Đây là một phần tử đa dụng mà thường được sử dụng thường xuyên để tạo một khối chứa các phần tử HTML khác. Đây là một thẻ khối rất quan trọng đóng vai trò lớn trong việc tạo nhóm các thẻ HTML khác và áp dụng CSS trong nhóm các phần tử. Thẻ có thể được sử dụng để tạo ra bố cục (layout) cho Web, tại đây nhà thiết kế xác định các phần khác nhau (trái, phải, trên,…) của trang bằng cách sử dụng thẻ này. Thẻ này không cung cấp bất cứ sự thay đổi về thị giác trên khối nhưng nó có nhiều ý nghĩa hơn khi được đặt tên (bằng cách đặt id hoặc class) và sử dụng cùng với các thuộc tính CSS.

Dưới đây là một ví dụ đơn giản về việc sử dụng thẻ <**div**> để tạo ra một phần tử khối:

```
<div>
  <p><em>Ngôn ngữ HTML là gì?</em></p>
  HTML viết tắt của <strong>Hypertext Markup Language</strong> là ngôn ngữ lập
trình dùng để xây
  dựng và cấu trúc lại các thành phần có trong Website...
</div>
```

2.5.2 Phần tử nội tuyến

Các **phần tử nội tuyến** là các phần tử mặc định sẽ xuất hiện cùng nhau trên dòng. Các phần tử nội tuyến được tạo bởi các thẻ **, <b>, <i>, <u>, , , <sup>, <sub>, <big>, <small>, <li>, <ins>, <del>, <code>, <cite>, <dfn>, <kbd>, <var>**… Thẻ <**span**> là một phần tử nội tuyến tiêu biểu và được sử dụng tương đối phổ biến. Thẻ này cũng không cung cấp bất cứ sự thay đổi về thị giác trên khối, nhưng nó có nhiều ý nghĩa hơn khi sử dụng cùng với CSS. Sự khác nhau giữa thẻ **** và thẻ ** ** là thẻ **** được sử dụng với các phần tử nội tuyến (chứa các phần tử nội tuyến khác bên trong) trong khi thẻ ** ** thường được sử dụng với cả phần tử nội tuyến và phần tử khối. Dưới đây là ví dụ đơn giản về thẻ ****:

```
<div>
  Đây là bảng tóm tắt tất cả các <span style="font-weight: bold">thuộc tính được
sử dụng trong <em style="color: #1abc9c">CSS</em></span>. Sẽ rất cần thiết cho
bạn tham
  khảo,tra cứu khi sử dụng CSS. Chi tiết từng thuộc tính sẽ giới thiệu ở từng
bài riêng...
</div
```

Kết quả hiển thị trên trình duyệt:

2.6 PHẦN TỬ NGỮ NGHĨA

HTML cung cấp các phần tử ngữ nghĩa để xác định các phần khác nhau của trang Web. Một phần tử ngữ nghĩa mô tả rõ ràng ý nghĩa của nó đối với cả máy tính và con người. Ví dụ các phần tử **<main><article>, <form>, < table >, <figcaption>, <figure>**... Mô tả nội dung bên trong các phần tử này chứa là gì. Tuy nhiên có những phần tử không có ngữ nghĩa. Ví dụ về các phần tử phi ngữ nghĩa như ** **, **** không nói gì về nội dung của nó (trừ khi nó được đặt id hoặc class). Sử dụng các phần tử ngữ nghĩa khiến các công cụ tìm kiếm có thể dễ dàng xác định đúng nội dung trang Web. Dưới đây là một số phần tử ngữ nghĩa hay dùng:

Phần tử <**main**> khai báo nội dung chính của tài liệu. Trong phần tử này có thể chứa các phần (section), bài viết (article), hoặc chứa các tiêu đề (heading), đoạn văn (paragraph) tùy theo mục đích của người thiết kế. Chú ý chỉ nên có một phần tử <main> trong tài liệu HTML, và phần tử này nên đặt ngang với các phần tử như **<header><nav> hay <footer>**. Ví dụ cấu trúc của Website:

```
<body>
    <header>
    </header>
    <div class="navbar">
    </div>
    <section>
    </section>
    <main>
    </main>
    <footer>
    </footer>
</body>
```

Đoạn mã trên có thể sẽ mô tả một Website với bố cục như sau:

<header>	
<section>	<main>
<footer>	

Phần tử <**section**> xác định một nhóm nội dung theo chủ đề. Ví dụ một <**section**> nhóm các nội dung về tên, hình ảnh thông tin liên hệ của tác giả:

```
<section>
  <h1>Phạm Dương Minh</h1>
  <img src="images/me.jpg" height="200px" class="avatar" />
  <q style="display: block">"Tạo ra những sản phẩm tốt nhất, được mọi người đón
nhận và sử dụng chính là đam
    mê của tôi"</q>
  <div class="contact">
    <h4 class="contact-title">Hòm thư</h4>
    <a href="mailto:(Webmail@gmail.com)">minh@gmail.com</a>
    <h4 class="contact-title">Điện thoại</h4>
    <a href="tel:+1255-568-6523">+1255 - 568 - 6523</a>
    <h4 class="contact-title">Địa chỉ</h4>
    <a href="https://www.google.com/maps" target="blank">Bowery St, New York, NY
10013,USA</a>
  </div>
</section>
```

Kết quả hiển thị của <**section**> trên như sau:

Hình 2-11 Một <section> thể hiện thông tin liên hệ

Phần tử <**article**> xác định một bài viết, đó có thể là bài viết trên một nhật ký hoặc một tạp chí. Một trang có thể có nhiều bài viết và thường có chung một cấu trúc. Ví dụ:

```
<article>
  <h2>Bảng các thuộc tính CSS</h2>
  <h3>Ngày đăng: <time>21/10/2022</time></h3>
  <content>
    <img src="images/css.png"></img>
    <div>
      Đây là bảng tóm tắt tất cả các <span style="font-weight: bold">thuộc tính
được sử dụng trong <em style="color: #1abc9c">CSS</em></span>. Sẽ rất cần thiết
cho bạn tham
      khảo,tra cứu khi sử dụng CSS. Chi tiết từng thuộc tính sẽ giới thiệu ở
từng bài riêng...
    </div>
  </content>
  <a class="readmore">Đọc thêm</a>
</article>
```

Trong một bài viết có thể có phần nội dung chi tiết, có phần tổng kết, kết luận. Những nội dung này cũng có thể được khai báo trong tài liệu HTML bằng cách sử dụng các phần tử như <**details**> và <**summary**>.

Phần tử <**header**> được sử dụng để chứa phần đầu của một cái gì đó, có thể là phần đầu của trang Web, hoặc đơn giản là phần đầu của một bài viết. Ví dụ sau định nghĩa phần đầu cho một bài viết chứa các thông tin về tiêu đề của bài viết đó:

```
<article>
 <header>
  <h1>What Does WWF Do?</h1>
  <p>WWF's mission:</p>
 </header>
 <p>WWF's mission is to stop the degradation of our planet's natural
environment,
 and build a future in which humans live in harmony with nature.</p>
</article>
```

Phần tử <**footer**> thường được sử dụng để chứa các thông tin tới tác giả của tài liệu, thông tin bản quyền, liên kết đến các điều khoản sử dụng, thông tin liên hệ, v.v. Phần chân này có thể là phần chân của cả tài liệu hoặc đơn giản là chân của một bài viết, hoặc một nhóm bài viết. Ví dụ sau đây sử dụng phần tử footer để khai báo phần chân cho một bài viết:

```
<article>
...
<footer>
   <p>Đăng bởi: Trần Tuấn Anh</p>
   <p>Liên hệ <a href="mailto:tuan_anh@gmail.com">tuan_anh@gmail.com</a></p>
</footer>
</article>
```

Phần tử <**nav**> xác định một khu vực điều hướng. Bên trong phần tử **<nav>** thường chứa các liên kết để phục vụ cho việc điều hướng. Ví dụ:

```
<nav>
   <a href="/html/">HTML</a> |
   <a href="/css/">CSS</a> |
   <a href="/js/">JavaScript </a> |
   <a href="/jquery/">jQuery</a>
</nav>
```

Phần tử <**aside**> xác định một số nội dung ngoài nội dung chúng của tài liệu (như một thanh bên). Các <**aside**> thường là các nội dung liên quan đến nội dung chính.

Ví dụ:

```
<p>My family and I visited The Epcot center this summer.</p>
<aside>
   <h4>Epcot Center</h4>
   <p>The Epcot Center is a theme park in Disney World, Florida.</p>
</aside>
```

<figure> kết hợp với phần tử **<figcaption>** được sử dụng để nhóm ảnh với tiêu đề của nó. Ví dụ:

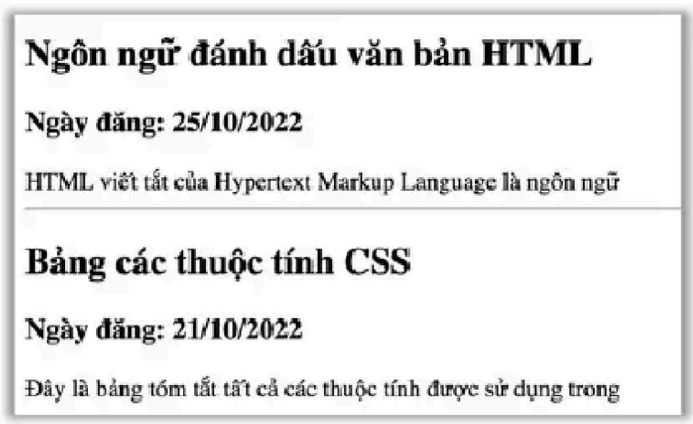
```
<figure>
   <img src="pic_trulli.jpg" alt="Trulli">
   <figcaption>Fig1. - Trulli, Puglia, Italy.</figcaption>
```

```
</figure>
```

Phần tử **<hr>** xác định ngắt theo chủ đề trong trang HTML và thường được hiển thị dưới dạng một dòng ngang. Phần tử <**hr**> được sử dụng để tách nội dung hoặc xác định một sự thay đổi trong một trang HTML. Thẻ <**hr**> xác định ngắt theo chủ đề trong trang HTML và thường được hiển thị dưới dạng quy tắc ngang. Phần tử <hr> được sử dụng để tách nội dung hoặc xác định một sự thay đổi trong một trang HTML. Ví dụ:

```
<article>
  <h2>Ngôn ngữ đánh dấu văn bản HTML</h2>
  <h3>Ngày đăng: 25/10/2022</h3>
  <content>
      HTML viết tắt của Hypertext Markup Language là ngôn ngữ
  </content>
</article>

<hr>

<article>
  <h2>Bảng các thuộc tính CSS</h2>
  <h3>Ngày đăng: 21/10/2022</h3>
  <content>
        Đây là bảng tóm tắt tất cả các thuộc tính được sử dụng trong
  </content>
</article>
```

Kết quả hiển thị của thẻ **hr** là một đường gạch ngang như dưới đây (có thể thay đổi màu sắc, độ dày… của đường theo ý muốn):



Lưu ý các phần tử có thể lồng nhau, ví dụ phần tử <**section**> xác định nhóm thông tin trong tài liệu. Trong <**section**> có thể chưa các phần tử <**article**>, các phần tử <**article**> này cũng có thể chứa các phần tử <**section**> bên trong. Lưu ý các phần tử ngữ nghĩa được sử dụng một cách linh động, tuy nhiên không nên sử dụng quá nhiều các cấu trúc lồng nhau khiến cho việc phân tích cú pháp để hiểu được nội dung của tài liệu trở lên khó khăn.

2.7 VĂN BẢN

2.7.1 Tiêu đề

Các tiêu đề và và các đoạn văn là những thành phần rất phổ biến trong các trang Web. Ví dụ một Website tin tức thì sẽ có tiêu đề của bài viết, hoặc trong một Website bán hàng thì tiêu đề có thể là tên của sản phẩm. Các tiêu đề luôn chứa các nội dung quan trọng. Một tài liệu có thể có nhiều mức tiêu đề và trong một trang Web chúng được xác định bằng các thẻ từ <**h1**> đến <**h6**>. Các thẻ này sẽ tạo ra các phần tử thuộc vào lại phần tử khối. Tức là mặc định nó sẽ chiếm toàn bộ độ rộng của trang Web. Các phần tử <**h1**> (tiêu đề mức 1) sẽ chứa nội dung quan trọng nhất, trong khi các phần tử <**h2>** (tiêu đề mức 2) sẽ dùng cho các nội dung ít quan trọng hơn, tiếp đến là <**h3>** tới <**h6**> (tiêu đề mức 3, 4, 5, 6) sẽ giảm dần về mức độ quan trọng. Mỗi tiêu đề HTML đều có kích thước mặc định thể hiện trong bảng sau:

Bảng 2-2 Mô tả các thuộc tính tiêu đề từ h1 đến h6

Phần tử	Mô tả
<**h1**>	Tạo ra tiêu đề mức 1, chữ đậm, có cỡ chữ bằng 200% cỡ chữ bình thường
<**h2**>	Tạo ra tiêu đề mức 2, chữ đậm, có cỡ chữ bằng 150% cỡ chữ bình thường
<**h3**>	Tạo ra tiêu đề mức 3, chữ đậm, có cỡ chữ bằng 117% cỡ chữ bình thường
<**h4**>	Tạo ra tiêu đề mức 4, chữ đậm, có cỡ chữ bằng 100% cỡ chữ bình thường
<**h5**>	Tạo ra tiêu đề mức 5, chữ đậm, có cỡ chữ bằng 83% cỡ chữ bình thường
<**h6**>	Tạo ra tiêu đề mức 6, chữ đậm, có cỡ chữ bằng 67% cỡ chữ bình thường

Ví dụ đoạn mã sau thể hiện đầu ra của các phần tử h1 tới h6 so với cỡ chữ bình thường:

```
Cỡ chữ bình thường
<h1>Tiêu đề mức 1</h1>
<h2>Tiêu đề mức 2</h2>
<h3>Tiêu đề mức 3</h3>
<h4>Tiêu đề mức 4</h4>
<h5>Tiêu đề mức 5</h5>
<h6>Tiêu đề mức 6</h6>
```

Kết quả hiển thị trên trình duyệt:

Hình 2-12 Tiêu đề từ h1 đến 6 khi hiển thị trên trình duyệt

Tuy nhiên, có thể chỉ định kích thước cho bất kỳ tiêu đề nào có thuộc tính style, sử dụng thuộc tính kích thước phông chữ CSS. Ví dụ: chỉ định cho tiêu đề <**h1**> có kích thước cỡ chữ là **60px** như sau:

```
<h1 style="font-size:60px;">Heading 1</h1>
```

Trình duyệt tự động thêm một khoảng trắng cách lề (margin) trước và sau tiêu đề. Công cụ tìm kiếm sử dụng các tiêu đề để lập chỉ mục cấu trúc và nội dung của các trang Web. Người dùng lướt qua các trang và thường đọc phần tiêu đề trước sau đó đọc các tiêu đề phụ và nội dung. Điều quan trọng là sử dụng các tiêu đề để hiển thị cấu trúc tài liệu một cách hợp lý. Ví dụ:

```
<h1>BÀI VIẾT</h1>
<article>
  <h2>Ngôn ngữ đánh dấu văn bản HTML</h2>
  <h3>Ngày đăng: <time>25/10/2022</time></h3>
  <content>
    <img src="images/html.jpg"></img>
    <div>
      <p><em>Ngôn ngữ HTML là gì?</em></p>
      HTML viết tắt của <strong>Hypertext Markup Language</strong> là ngôn ngữ
lập trình dùng để xây
      dựng và cấu trúc lại các thành phần có trong Website...
    </div>
  </content>
  <a class="readmore">Đọc thêm</a>
</article>
```

Kết quả hiển thị trên trình duyệt:

Hình 2-13 Ví dụ sử dụng tiêu đề <h1> và <h2> trên một bài viết

Trong ví dụ trên, tiêu đề <**h1**> được sử dụng cho các tiêu đề chính, tiếp theo là tiêu đề <**h2**> ít quan trọng hơn, v.v... Các tiêu đề luôn có chữ đậm và lớn hơn chữ thường. Tuy nhiên, trong thiết kế, chỉ sử dụng các phần tử <**h1**> đến <**h6**> này cho tiêu đề mà không sử dụng để tạo các chữ lớn hoặc in đậm. Việc định dạng về mặt hiển thị như cỡ chữ, độ đậm của font chữ là công việc của các thuộc tính **CSS**.

2.7.2 Đoạn văn

Một tài liệu HTML, đặc biệt là các tài liệu chứa các bài viết dài thường có nhiều đoạn văn. Các đoạn văn được tạo ra bởi các phần tử <**p**> (paragraph). Các đoạn văn cũng giống phần tử tiêu đề, đó là một phần tử khối, tức là mặc định sẽ chiếm toàn bộ chiều rộng của tài liệu. Ví dụ tạo ra 2 đoạn văn:

```
<p><em>Ngôn ngữ HTML là gì?</em></p>
<p>HTML viết tắt của <strong>Hypertext Markup Language</strong> là ngôn ngữ
lập trình dùng để xây dựng và cấu trúc lại các thành phần có trong
Website...</p>
```

Trình duyệt tự động thêm một số khoảng trắng trước và sau một đoạn. Tuy nhiên với mỗi trình duyệt hoặc tùy vào kích thước trình duyệt có thể có kết quả hiển thị khác nhau. Để thêm các khoảng trắng này cần sử dụng các thuộc tính **margin** và **padding** trong CSS. Vì việc thêm khoảng trắng, hoặc các dấu xuống dòng trong mã HTML là không có ý nghĩa. Trình duyệt sẽ xóa mọi **khoảng trắng thừa** và các **dòng thừa** khi trang được hiển thị. Ví dụ, khi viết mã như dưới đây thì những chỗ xuống dòng khi viết mã lệnh sẽ không có ý nghĩa và không được hiển

thị trên trình duyệt. Ví dụ dấu xuống dòng sau chữ "ngôn ngữ: ở ví dụ trên sẽ không khiến các ký tự sau xuất hiện trên một dòng mới. Nếu người thiết kế chủ đích muốn xuống dòng, cần sử dụng tới phần tử ngắt dòng <**br**> (break).

Đối với một số đoạn văn đặc biệt như trích dẫn, ngoài thẻ <**p**>, có thể sử dụng các thẻ <**blockquote**>. Phần tử <**blockquote**> xác định một nội dung được trích dẫn từ một nguồn khác. Các trình duyệt mặc định sẽ thụt lề các nội dung trong phần tử <**blockquote**>. Ví dụ:

```
Đây là một trích dẫn từ Trang Web của WWF:
<blockquote cite="http://www.worldwildlife.org/who/index.html">
 WWF hoạt động ở 100 quốc gia và được hỗ trợ bởi 1,2 triệu thành viên ở Hoa Kỳ
và gần 5 triệu trên toàn cầu với mục tiêu bảo vệ thiên nhiên
</blockquote>
```

Thẻ <**q**> được sử dụng cho các trích dẫn ngắn. Các trình duyệt thường chèn dấu ngoặc kép cho những trích dẫn này. Phần tử <**q**> là một phần tử thuộc loại nội tuyến, chính vì vậy nó sẽ nằm cùng dòng với các phần tử nội tuyến khác, vì vậy nếu muốn nội dung này xuất hiện trên một dòng, cần đổi **display** thành **block**. Ví dụ:

```
<img src="images/me.jpg" height="200px" class="avatar" />
<q style="display: block">Nếu không có đam mê, sẽ không có năng lượng</q>
```

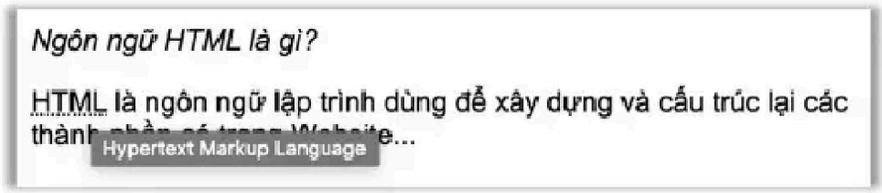
Đoạn mã trên sẽ hiển thị như hình dưới đây. Có thể thấy rằng nhờ có thiết lập display:block mà đoạn văn bản trong phần tử <**q**> không nằm trên cùng một dòng với bức ảnh:

Hình 2-14 Phần tử <q> dùng để hiển thị một câu nói hoặc trích dẫn

Phần tử <**abbr**> khai báo một từ (hoặc cụm từ) viết tắt. Việc đánh dấu đây là một từ viết tắt rất có giá trị cho các hệ thống tìm kiếm, khi đó dữ liệu có thể được tìm kiếm bằng từ khóa viết tắt hoặc từ viết đầy đủ. Hầu hết các trình duyệt sẽ hiển thị từ đầy đủ khi người dùng di chuột tới từ viết tắt. Ví dụ:

```
<p><em>Ngôn ngữ HTML là gì?</em></p>
<p><abbr title="Hypertext Markup Language">HTML</abbr> là ngôn ngữ lập trình
dùng để xây dựng và cấu trúc lại các thành phần có trong Website...</p>
```

Kết quả hiển thị trên sẽ như sau:



Hình 2-15 Sử dụng phần tử <abbr> để mô tả một từ viết tắt

Phần tử <**address**> xác định thông tin liên hệ (tác giả / chủ sở hữu) của một nội dung. Phần tử <address> thường được hiển thị bằng chữ nghiêng.

```
<p>Đây là bảng tóm tắt tất cả các thuộc tính được sử dụng trong CSS. Sẽ rất cần
thiết cho bạn tham khảo,tra cứu khi sử dụng CSS. Chi tiết từng thuộc tính sẽ
giới thiệu ở từng bài riêng...</p>
<address>
  Biên tập bởi Dương Minh<br> Trang chủ: https://minh-blogs.Web.app
</address>
```

Phần tử <**cite**> dùng để tham chiếu tới một sản phẩm nào đó (thường là nổi tiếng hoặc được biết đến) ví dụ như một cuốn sách, một bức tranh, một bài thơ.... Trình duyệt thường hiển thị các phần tử <cite> in nghiêng:

```
<p><cite>Tác phẩm tiếng thét (The Scream)</cite> được sáng tác bởi Edvard Munch
năm 1893</p>
```

Thẻ <**time**> - Được sử dụng để tạo nội dung chứa thông tin về thời gian. Ví dụ:

```
<article>
  <h2>Lập trình Javascript căn bản</h2>
  <h3>Ngày đăng: <time>1/12/2022</time></h3>
  <content>
  ...
  </content>
</article>
```

Bên cạnh những thẻ HTML nói trên, có rất nhiều thẻ khác liên quan đến văn bản. Ví dụ đối với những văn bản sử dụng các ngôn ngữ không phải latinh, người thiết kế có thể phải quan đến việc sử dụng các phần tử như <**ruby**>, <**rt**>. Các nhà thiết kế Web cần phải tìm hiểu thêm trong quá trình thiết kế và thường xuyên cập nhật những những phần tử HTML mới xuất hiện trong các phiên bản HTML được ra mắt.

2.7.3 Định dạng văn bản

Trong văn bản, có một số nội dung quan trọng muốn nhấn mạnh để người dùng tập trung hơn, có thể sử dụng một số phần tử định dạng. Ví dụ, như phần tử <**b**> sẽ khiến cho chữ đậm hơn (bold) hoặc phần tử <**i**> sẽ khiến chữ biến thành chữ nghiêng (italic). Trong quá khứ cách định dạng văn bản bằng cách sử dụng các phần tử này rất phổ biến. Tuy nhiên, ngày nay việc định dạng về mặt hiển thị thì các nhà thiết kế ưu tiên sử dụng các thuộc tính **CSS** (sẽ đề cập trong phần sau). Một số thẻ để định dạng văn bản có thể kể đến như:

- **<b>** - Chữ in đậm
- **** - Chữ chứa nội dung quan trọng
- **<i>** - Chữ in nghiêng
- **** - Nội dung nhấn mạnh trong văn bản
- **<mark>** - Đánh dấu văn bản
- **<small>** - Văn bản chữ nhỏ
- <**del**> - Văn bản bị xóa
- <**s**> - Văn bản không còn đúng nữa
- <**u**> - Văn bản viết sai (chính tả)
- **<ins>** - Văn bản chèn thêm
- **<sub>** - Chỉ số dưới
- **<sup>** - Chỉ số trên

Ví dụ, một văn bản với chữ đậm sử dụng thẻ <**b>** và một văn bản khác được nhấn mạnh sử dụng thẻ **strong**:

```
<b>Văn bản sử dụng thẻ b</b>
<strong>Văn bản sử dụng thẻ strong</strong>
```

Kiểm tra kết quả hiển thị của 2 phần tử nói trên ở trình duyệt có thể nhận thấy rằng phần tử <**b**> và <**strong**> cho kết quả hiển thị giống nhau. Tuy nhiên, cần phải hiểu bản chất là phần tử <**b**> xác định văn bản in đậm mà không có bất kỳ tầm quan trọng nào. Trong khi đó, phần tử <**strong**> xác định một nội dung quan trọng về mặt ngữ nghĩa. Tương tự, phần tử <**i>** xác định văn bản in nghiêng trong khi phần tử <**em**> xác định văn bản được nhấn mạnh. Điều này giúp cho máy tính hoặc các hệ thống tìm kiếm biết được đâu là những nội dung quan trọng trong tài liệu HTML.

Thẻ **<del>** định nghĩa văn bản đã xóa. Thẻ **<ins>** xác định văn bản được chèn thêm nội dung. Có phần tử này cho biết dữ liệu vừa được thêm vào hoặc xóa khỏi văn bản. Ví dụ khi muốn thể hiện màu sắc **blue** đã bị thay thế bởi màu **red** trong một đoạn văn bản, hoặc thể hiện từ **color** mới được thêm vào câu:

```
<p>My favorite color is <del>blue</del> red.</p>
<p>My favorite <ins>color</ins> is red.</p>
```

Phần tử **<sub>** HTML xác định chỉ số dưới của văn bản. Trong khi đó, phần tử <**sup**> HTML xác định chỉ số trên của văn bản.

```
<p>This text contains <sup>superscript</sup> text.</p>
<p>This text contains <sub>subscript</sub> text.</p>
```

2.7.4 Ngắt dòng

Phần tử HTML **
** xác định việc ngắt dòng trong văn bản. Có thể sử dụng thẻ <**br**> nếu muốn ngắt dòng (một dòng mới) mà không bắt đầu một đoạn mới. Ví dụ:

```
<p>Một đoạn văn bản sửa dụng <br> ngắt dòng</p>
```

Lưu ý, thẻ <**br**> là một thẻ trống, có nghĩa là nó không có thẻ kết thúc.

2.7.5 Văn bản mã lệnh (code)

Khi muốn biểu một mã lệnh máy tính (code) trên trang Web, có thể sử dụng thẻ <**code**>. Khi đó đoạn văn bản sẽ được hiểu là một đoạn mã lệnh và trình duyệt sẽ sử dụng font chữ phù hợp để biểu diễn đoạn mã này (thường là font chữ monospace). Khi đó các chữ cái sẽ có độ rộng bằng nhau. Ví dụ:

```
<code>
class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
</code>
```

Tuy nhiên khi hiển thị, các mã lệnh này sẽ bị hiển thị trên một dòng, để các mã lệnh hiển thị trên nhiều dòng có thể bổ sung thêm phần tử <**br**>. Tuy nhiên, để điều này được thực hiện dễ dàng hơn có thể sử dụng thẻ <**pre**> để bao nội dung trên. Phần tử <**pre**> khai báo một nội dung đã được định dạng xuống dòng rồi (pre format). Ví dụ nếu viết:

```
<code>
<pre>
class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
</pre>
</code>
```

Kết quả hiển thị trên trình duyệt là:

```
class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

Hình 2-16 Hiển thị mã lệnh trên trang Web

Chú ý khi biểu diễn các mã lệnh, có thể sử dụng thêm các phần tử sau để khai báo thêm thông tin cho đoạn mã lệnh:

Phân tử	Mô tả
<samp>	Kết quả đầu ra của đoạn mã lệnh

<kbd>	Bàn phím nhập đầu vào
<var>	Khai báo biến

2.8 BẢNG BIỂU

Các bảng cho phép sắp xếp các dữ liệu như văn bản, hình ảnh, đường link… vào các ô trong bảng. Để khai báo thông tin một bảng trong tài liệu HTML, sử dụng các phần tử <**table**>, <**tr**>, <**td**>, <**th**>, <**tbody**> ...

2.8.1 Thẻ Table

Bảng trong tài liệu HTML được tạo ra bằng cách sử dụng thẻ <**table**>**.** Trong đó thẻ <**tr**> được sử dụng để tạo các dòng (table row) và thẻ <**td**> được sử dụng để tạo các ô dữ liệu (table datum).

Ví dụ bảng trong HTML:

```
<table>
    <tr>
        <th>Thuộc tính</th>
        <th>Mô tả</th>
    </tr>
    <tr>
        <td>accent-color</td>
        <td>Thuộc tính xác định màu sắc của các điều khiển</td>
    </tr>
    <tr>
        <td>align-content</td>
        <td>Thuộc tính xác định căn chỉnh cho các item trong một hộp chứa</td>
    </tr>
</table>
```

Chạy đoạn mã trên sẽ tạo bảng sau:

Thuộc tính	Mô tả
accent-color	Thuộc tính xác định màu sắc của các điều khiển
align-content	Thuộc tính xác định căn chỉnh cho các item trong một hộp chứa

Các hàng được định nghĩa bởi thẻ <**tr**> và các cột dữ liệu được định nghĩa bằng thẻ <**td**>. Tiêu đề bảng có thể được xác định bằng thẻ <**th**>. Thẻ này đại diện cho các cột dữ liệu. Thông thường sẽ đặt hàng đầu tiên của bảng là tiêu đề như hình, ngoài ra cũng có thể sử dụng phần tử <**th**> trong bất kỳ hàng nào. Có thể thấy rằng bảng sẽ giúp chia các phần tử theo hàng cột, tuy nhiên mặc định thì bảng sẽ không có các viền. Để thêm viền cho bảng và các hàng cột, sử dụng thuộc tính **style** với việc khai báo viền **border** như sau:

```
<table style="border-collapse: collapse;">
    <tr>
        <th style="border: 1px solid black">Thuộc tính</th>
        <th style="border: 1px solid black">Mô tả</th>
    </tr>
    <tr>
        <td style="border: 1px solid black">accent-color</td>
        <td style="border: 1px solid black">Thuộc tính xác định màu sắc của các
điều khiển</td>
```

```
    </tr>
    <tr>
        <td style="border: 1px solid black">align-content</td>
        <td style="border: 1px solid black">Thuộc tính xác định căn chỉnh cho
các item trong một hộp chứa</td>
    </tr>
</table>
```

Kết quả hiển thị như sau:

Thuộc tính	Mô tả
accent-color	Thuộc tính xác định màu sắ của các điều khiển
align-content	Thuộc tính xác định căn chỉnh cho các item trong một hộp chứa

2.8.2 Thuộc tính colspan và rowspan

Sử dụng thuộc tính **colspan** để nhập hai hay nhiều cột vào một cột. Tương tự là thuộc tính **rowspan** để nhập hai hay nhiều hàng vào một hàng. Ví dụ:

```
<table>
   <tr>
       <th>Column 1</th>
       <th>Column 2</th>
       <th>Column 3</th>
   </tr>
   <tr><td rowspan="2">Row 1 Cell 1</td><td>Row 1 Cell 2</td><td>Row 1 Cell
3</td></tr>
   <tr><td>Row 2 Cell 2</td><td>Row 2 Cell 3</td></tr>
   <tr><td colspan="3">Row 3 Cell 1</td></tr>
</table>
```

Kết quả hiển thị :

Column 1	Column 2	Column 3
Row 1 Cell 1	Row 1 Cell 2	Row 1 Cell 3
	Row 2 Cell 2	Row 2 Cell 3
Row 3 Cell 1		

2.8.3 Tiêu đề cho bảng

Các bảng có thể có tiêu đề bằng cách bổ sung thêm phần tử <**caption**>. Các tiêu đề này là tên của bảng hoặc được sử dụng để ghi lời giải thích cho bảng. Mặc định các tiêu đề này sẽ xuất hiện ở đầu bảng:

```
<table>
<caption>Tiêu đề</caption>
   <tr>
       <td>Cột 1, Hàng 1</td><td>Cột 2, Hàng 1</td>
   </tr>
   <tr>
       <td>Cột 1, Hàng 2</td><td>Cột 2, Hàng 2</td>
```

```
   </tr>
</table>
```

Kết quả:

Tiêu đề

Cột 1, Hàng 1	Cột 2, Hàng 1
Cột 1, Hàng 2	Cột 2, Hàng 2

Để tiêu đề xuất hiện ở các vị trí khác, cần khai báo thêm thuộc tính, caption-side, ví dụ muốn tiêu đề xuất ở cuối của bảng:

```
<caption style="caption-side:bottom">Tiêu đề</caption>
```

2.8.4 Đầu, thân và chân bảng trong HTML

Bảng có thể được chia làm 3 phần: phần đầu, phần thân chứa dữ liệu và một phần chân bảng chứa các nội dung tổng kết hoặc ghi chú. Đầu và chân bảng thì tương tự như đầu trang và chân trang trong các tài liệu văn bản, trong khi phần thân là nội dung chính được hiển thị trong bảng. Các phần này được khai báo bởi các thẻ sau:

- <**thead**> - phần đầu bảng
- <**tbody**> - phần thân bảng
- <**tfoot**> - phần chân bảng

Ví dụ:

```
<table >
   <thead>
   <tr>
       <td colspan="4">Day la phan tieu de bang</td>
   </tr>
   </thead>
   <tbody>
   <tr>
       <td>Cell 1</td>
       <td>Cell 2</td>
       <td>Cell 3</td>
       <td>Cell 4</td>
   </tr>
   </tbody>

   <tfoot>
   <tr>
       <td colspan="4">Day la phan chan bang</td>
   </tr>
   </tfoot>
</table>
```

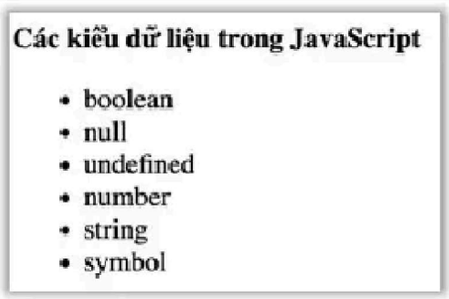
2.9 DANH SÁCH

2.9.1 Danh sách không có thứ tự

Danh sách không có thứ tự là danh sách không được đánh số thứ tự, danh sách này thường các phần tử sẽ có một dấu ký hiệu đầu dòng như một vòng tròn nhỏ màu đen (theo mặc định). Để tạo ra một danh sách không có thứ tự sử dụng thẻ <**ul**> (Unorderd list) kết hợp với các phần tử <**li**> (List item)**.** Ví dụ:

```
<strong>Các kiểu dữ liệu trong JavaScript</strong>
<ul>
    <li>boolean</li>
    <li>null</li>
    <li>undefined</li>
    <li>number</li>
    <li>string</li>
    <li>symbol</li>
</ul>
```

Kết quả hiển thị:



Hình 2-17 Danh sách không có thứ tự với các dấu chấm đầu dòng

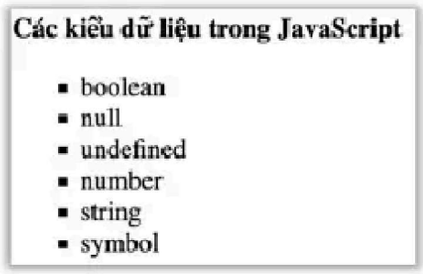
Có thể sử dụng thuộc tính **style** cho phần tử <**ul**> để xác định kiểu của ký hiệu đầu dòng. Theo mặc định nó có hình dạng đĩa tròn (disc). Có các tùy chọn kiểu khác có thể sử dụng:

- list-style-type: square
- list-style-type:disc
- list-style-type:circle

Ví dụ:

```
<strong>Các kiểu dữ liệu trong JavaScript</strong>
<ul style="list-style-type:square">
    <li>boolean</li>
    <li>null</li>
    <li>undefined</li>
    <li>number</li>
    <li>string</li>
    <li>symbol</li>
</ul>
```

Kết quả :



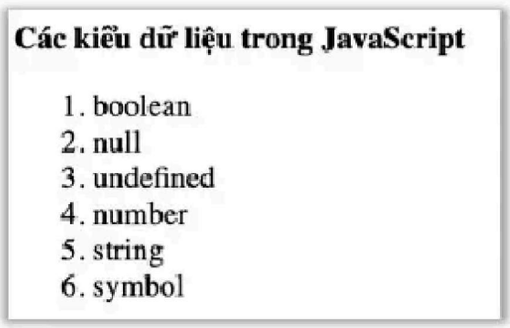
Hình 2-18 Danh sách với các dấu vuông đầu dòng

2.9.2 Danh sách có thứ tự

Một danh sách có thể đánh thứ tự theo số hoặc theo bảng chữ cái, khi đó sẽ được danh sách có thứ tự. Danh sách này được tạo bởi thẻ <**ol**>. Mặc định thứ tự bắt đầu từ số 1, và cũng được tạo bởi các phần tử <**li**>. Ví dụ:

```
<strong>Các kiểu dữ liệu trong JavaScript</strong>
<ol>
    <li>boolean</li>
    <li>null</li>
    <li>undefined</li>
    <li>number</li>
    <li>string</li>
    <li>symbol</li>
</ol>
```

Kết quả tạo ra như sau:



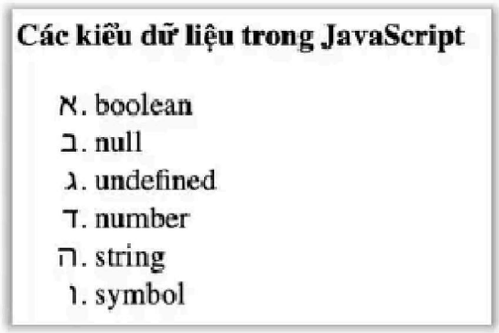
Một danh sách thử tự cũng có thể được đánh số với nhiều kiểu khác nhau như:

- list-style-type: armenian
- list-style-type: cjk-ideographic
- list-style-type: decimal
- list-style-type: decimal-leading-zero
- list-style-type: georgian
- list-style-type: hebrew

Ví dụ:

```
<strong>Các kiểu dữ liệu trong JavaScript</strong>
<ol style="list-style-type: hebrew">
    <li>boolean</li>
    <li>null</li>
    <li>undefined</li>
    <li>number</li>
    <li>string</li>
    <li>symbol</li>
</ol>
```

Kết quả hiển thị:



Có thể sử dụng thuộc tính start cho thẻ <**ol**> để xác định điểm bắt đầu của dãy số. Dưới đây là các tùy chọn có thể. Ví dụ:

```
<strong>Các kiểu dữ liệu trong JavaScript</strong>
<ol start="4">
    <li>boolean</li>
    <li>null</li>
    <li>undefined</li>
    <li>number</li>
    <li>string</li>
    <li>symbol</li>
</ol>
```

Kết quả:

2.9.3 Danh sách định nghĩa

HTML hỗ trợ một kiểu danh sách mà được gọi **Danh sách định nghĩa** là danh sách mà các mục được liệt kê dưới dạng giống một từ điển hoặc một quyển bách khoa toàn thư. Danh sách này là một cách rất tốt để hiển thị một **thực đơn** hoặc một **bảng chú giải** của các mục dữ liệu.

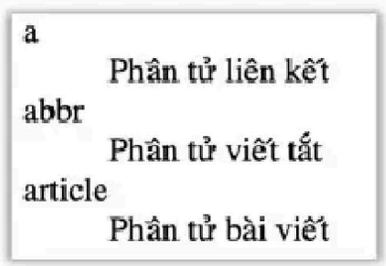
Danh sách định nghĩa sử dụng 3 thẻ theo sau:

- <dl> - Xác định danh sách (defination list)
- <dt> - Thuật ngữ (defination term)
- <dd> - Mô tả cho thuật ngữ đó (defination description)

Ví dụ:

```
<strong>Bảng tra cứu các phần tử HTML</strong>
<dl>
    <dt>a</dt>
    <dd>Phần tử liên kết</dd>
    <dt>abbr</dt>
    <dd>Phần tử viết tắt</dd>
    <dt>article</dt>
    <dd>Phần tử bài viết</dd>
</dl>
```

Kết quả:



Hình 2-19 Một danh mục được tạo bởi phần tử <dl>, <dt> và <dd>

2.10 LIÊN KẾT

Liên kết được sử dụng trong gần như tất cả các trang Web. Liên kết cho phép người dùng chuyển từ trang này sang trang khác. Khi di chuyển chuột qua một liên kết, mũi tên chuột sẽ biến thành một bàn tay.

2.10.1 Tạo liên kết

Các liên kết được xác định bằng thẻ **<a>**, trong đó sử dụng thuộc tính **href** để xác định địa chỉ đích của liên kết. Ví dụ khi muốn tạo liên kết tới trang chủ của Minh's Blogs:

```
<a href="https://minh-blogs.Web.app">Tới trang Minh's Blogs</a>
```

Lưu ý: Không cần dấu gạch chéo ở cuối địa chỉ.

2.10.2 Liên kết cục bộ

Ví dụ trên đã sử dụng một URL tuyệt đối (một địa chỉ Web đầy đủ). Liên kết cục bộ (liên kết đến cùng một trang Web) được chỉ định bằng một URL tương đối. Ví dụ:

```
<a href="article.html">Đọc thêm</a>
```

2.10.3 Thuộc tính target

Thuộc tính **target** xác định nơi để mở tài liệu được liên kết. Thuộc tính này có thể có một trong các giá trị sau:

- **_blank** mở tài liệu được liên kết trong cửa sổ hoặc tab mới.
- **_self** mở tài liệu được liên kết trong cùng một cửa sổ/tab khi được nhấp (mặc định).
- **_parent** mở tài liệu được liên kết trong toàn bộ cửa sổ.

Ví dụ:

```
<a href="https://www.vncodelab.com/" target="_blank">Vào trang Vncodelab</a>
```

2.10.4 Hình ảnh liên kết

Các liên kết có thể gắn vào một hình ảnh. Ví dụ:

```
<a href="article.html">
<img src="images/js.png" alt="Javascript">
</a>
```

2.10.5 Tiêu đề liên kết

Thuộc tính **title** xác định thông tin thêm về một phần tử liên kết. Thông tin thường được hiển thị dưới dạng văn bản chú giải công cụ khi chuột di chuyển qua phần tử. Thông tin này thường giúp hiển thị thông tin dưới dạng văn bản chú giải khi người dùng chuột di chuyển qua phần tử.

```
<a class="readmore" href="article.html" title="Đọc thêm về JavaScript">Đọc
thêm</a>
```

2.10.6 Liên kết đánh dấu

Đánh dấu được sử dụng để cho phép người đọc chuyển đến các phần cụ thể của trang Web. Đánh dấu có thể hữu ích nếu trang Web dài. Để thực hiện điều này, trước tiên phải tạo dấu trang và sau đó thêm một liên kết đến nó. Khi liên kết được nhấp, trang sẽ cuộn đến vị trí có dấu trang.

Để tạo dấu trang, tạo một phần tử với thuộc tính id:

```
<h2 id="C2">Chương 2</h2>
```

Sau đó, thêm một liên kết đến dấu trang ("Chuyển đến Chương 2"), từ trong cùng một trang:

```
<a href="#C2">Chuyển đến Chương 2</a>
```

Hoặc thêm một liên kết đến dấu trang ("Chuyển đến Chương 2"), từ một trang khác:

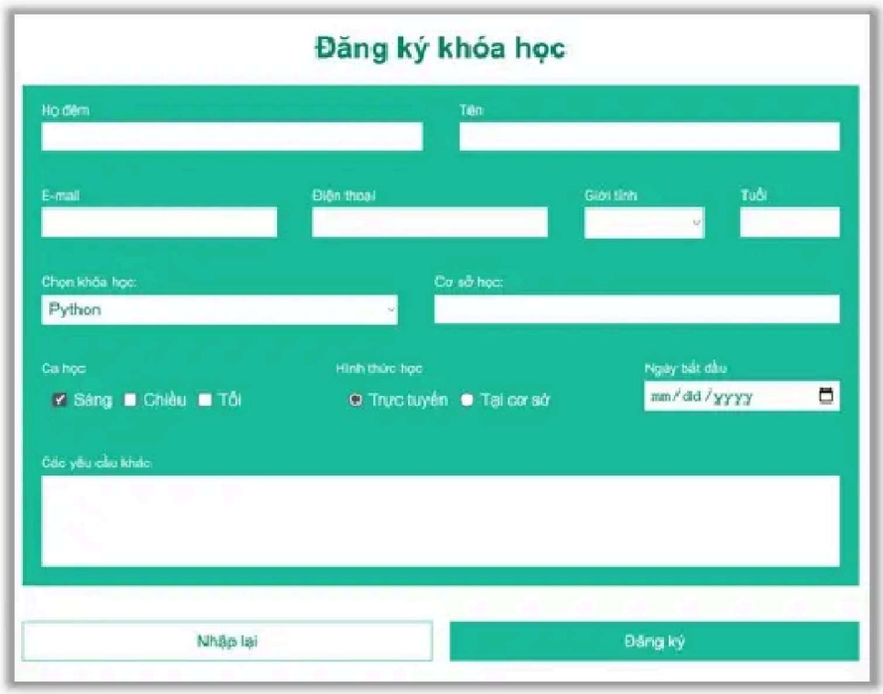
```
<a href="article.html#C42">Chuyển đến chương 2</a>
```

2.11 BIỂU MẪU

2.11.1 Biểu mẫu là gì?

Phần tử **<form>** định nghĩa ra **biểu mẫu (form)** được dùng để thu thập dữ liệu do người dùng nhập:

- Một phần tử **form** bao gồm các phần tử để người dùng **nhập** hoặc **lựa chọn** dữ liệu.
- Phần tử form có thể nhận nhiều kiểu thuộc tính đầu vào, như **văn bản** (text), **hộp kiểm** (checkbox), nút **lựa chọn** (radio), nút **bấm xác nhận** (submit),...
- Các phần tử của **form** hầu hết được tạo ra bởi các thẻ <**input**> với thuộc tính **type** xác định kiểu đầu vào khác nhau.
- Các phần tử form cũng được tạo ra bởi các thẻ **<textarea>**, **<select>**, **<option>**, **<button>** ...



Hình 2-20 Ví dụ một biểu mẫu đăng ký khóa học

Phần thử form được khai báo bằng thẻ **<form>**, ví dụ:

```
<form action="save.php">
   .
   Các phần tử nhập
   .
</form>
```

Phần tử <**form**> có nhiều thuộc tính như **action**, **method**,… Những thuộc tính này sẽ quyết định cách thức gửi dữ liệu từ máy người dùng tới Server. Trong <**form**> sẽ chứa các phần tử nhập. Phần tử nhập được tạo ra bởi thẻ **<input>**, **<textarea>**, **<select>**, **<option>**, **<button>**, có thể chia thành các nhóm như sau:

a. Văn bản

- <input type="text">
- <input type="email">
- <input type="number">
- <input type="range">
- <input type="search">
- <input type="tel">
- <input type="password">
- <input type="url">

b. Thời gian

- <input type="time">
- <input type="week">
- <input type="month">
- <input type="date">
- <input type="datetime-local">

c. Lựa chọn

- <input type="checkbox">
- <select>
- <input type="radio">
- <datalist>
- <option>

d. Nút bấm

- <input type="image">
- <input type="reset">
- <input type="submit">
- <input type="button">
- <button>

e. Các phần tử khác

- <input type="color">
- <input type="hidden">
- <input type="file">

2.11.2 Ô nhập văn bản

Để yêu cầu người dùng nhập văn bản đơn thuần (plain text), sử dụng phần tử <**input**> với type là **text**. Ví dụ:

```
<label for="last_name">Họ đệm:</label>
<input type="text" id="last_name" size="15">
<label class="form-sub-label" for="last_name">Tên:</label>
<input type="text" id="first_name" size="15">
```

Kết quả:

Hình 2-21 Ô nhập văn bản

Dữ liệu văn bản đơn thuần là kiểu dữ liệu phổ biến. Người sử dụng dùng để nhập liệu trên hệ thống, dữ liệu người dùng nhập có thể là dạng chuỗi, hoặc số, địa chỉ email, số điện thoại, dữ liệu này thường không được kiểm soát định dạng. Với một loại dữ liệu này HTML có đều có thuộc tính type tương ứng. Ví dụ với type là **email** được sử dụng cho trường đầu vào chứa một địa chỉ email.

```
<label for="email">E-mail:</label>
<input type="email" id="email">
```

Kết quả hiển thị:

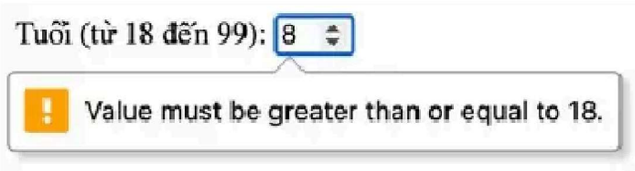
Hình 2-22 Ô nhập địa chỉ hòm thư

Tùy thuộc vào sự hỗ trợ của trình duyệt, địa chỉ **email** có thể tự động kiểm tra khi người dùng nhập hoặc gửi. Một số điện thoại thông minh có thể nhận ra kiểu **email** và thêm ".com" vào bàn phím ảo để hỗ trợ nhập **email** được nhanh hơn hoặc tự động gợi ý các email người dùng đã nhập trước đây.

Input với type là **number** định nghĩa trường đầu vào là số. Có thể thiết lập giới hạn với những số cho phép nhập. Ví dụ dưới đây hiển thị một trường đầu vào là số và chỉ có thể nhập giá trị từ 1-5:

```
<label for="age">Tuổi (từ 18 đến 99):</label> <input type="number" id="age"
min="18" max="99">
```

Kết quả khi người dùng một số ngoài khoảng từ 1 đến 5:



Hình 2-23 Ô nhập số

Ví dụ dưới đây hiển thị một trường số đầu vào, nơi có thể nhập giá trị từ 0-100, bước nhảy (step) là 10, giá trị ban đầu là 30:

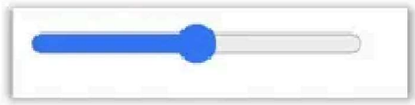
```
<input type="number" name="points" min="0" max="100" step="10" value="30">
```

Với ô nhập số, các trình duyệt thường hiển thị thêm mũi tên để giúp việc thay đổi giá trị được dễ dàng hơn. Ngoài ra, trên các thiết bị di động, bàn phím ảo này thường chuyển sang dạng bàn phím số để việc nhập số được thuận tiện. Dữ liệu trong các ô nhập số cũng được trình duyệt kiểm tra sau khi người dùng hoàn thành việc nhập liệu.

Ô nhập với với type là **range** định nghĩa một điều khiển để chọn một số không quan trọng sự chính xác. Mặc định, phạm vi các số được chọn là từ 0 đến 100. Tuy nhiên, cũng có thể tự thiết lập giá trị nhỏ nhất, lớn nhất, và bước thay đổi thông qua các thuộc tính như **min**, **max** và **step**. Ví dụ:

```
<input type="range" name="points" min="0" max="10">
```

Kết quả hiển thị:



Hình 2-24 Điều khiển để chọn giá trị số

Ô nhập với type là **search** được dùng cho các trường tìm kiếm. Ví dụ:

```
Tìm kiếm: <input type="search" name="googlesearch">
```

Kết quả hiển thị:

Hình 2-25 Ô nhập tìm kiếm

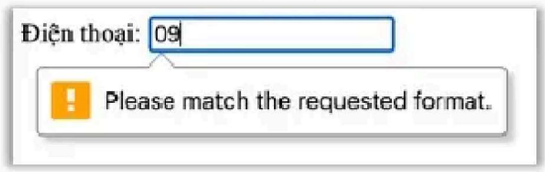
Ô nhập với type là **search** khác với **text** tùy thuộc vào từng trình duyệt, ví dụ một số trình duyệt sẽ hiển thị thêm dấu X để xóa những từ đang gõ. Hoặc hiển thị những từ tìm kiếm gần đây, những từ được tìm kiếm nhiều lần...

Ô nhập với type là **tel** được sử dụng để người dùng nhập số điện thoại, ví dụ:

```
<form>
   Telephone:
```

```
   <input type="tel" name="phone" pattern="[0-9]{3}-[0-9]{2}-[0-9]{3}">
</form>
```

Người dùng khi nhập vào:



Hình 2-26 Ô nhập số điện thoại

Khi sử dụng thuộc tính type là **tel**, ô nhập thường nhìn không khác với ô nhập văn bản đơn thuần, tuy nhiên khi sử dụng trên một số thiết bị di động, bàn phím ảo có thể tự động thay đổi để người dùng thuận tiện hơn cho việc nhập một số điện thoại, ví dụ chỉ hiện các ký tự cho phép như số, dấu cách, mã vùng,… Việc cung cấp một pattern cũng giúp dữ liệu được kiểm soát định dạng tốt hơn.

Ô nhập với type là **password** khiến cho nội dung người dùng nhập không hiển thị ra màn hình, mà thay bằng ký tự đại diện:

```
<form>
  Mật khẩu: <input type="password">
</form>
```

Ví dụ khi chạy với trình duyệt Chrome ký tự đại diện là hình tròn đen:

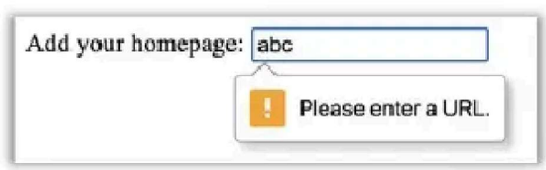
Hình 2-27 Ô nhập mật khẩu

Đối với các ô nhập như mật khẩu, các trình duyệt hiện nay thường ghi nhớ và tự động điền cho người dùng nếu đã xác thực được người dùng đó đúng là người sở hữu mật khẩu để truy cập, việc ghi nhớ mật khẩu người dùng có thể cấu hình trong các trình duyệt một cách dễ dàng.

Ô nhập với type là **url** được sử dụng cho trường đầu vào là một địa chỉ **url**:

```
<form>
   Add your homepage:
   <input type="url" name="homepage">
</form>
```

Kết quả:



Hình 2-28 Ô nhập địa chỉ URL

Tùy thuộc vào sự hỗ trợ của trình duyệt, trường **url** có thể tự động được kiểm tra và xác nhận khi người dùng nhập. Một số điện thoại thông minh có thể nhận dạng **url** và thêm ".com" vào bàn phím để phù hợp với đầu vào **url**.

2.11.3 Vùng nhập văn bản

Phần tử <**textarea**> xác định một vùng nhập nhiều dòng cho phép nhập văn bản trên một không gian rộng hơn, kích thước của vùng nhập văn bản được xác định bởi số dòng (thuộc tính **rows**) và số cột trên mỗi dòng (thuộc tính **cols**):

```
<label for="request">Các yêu cầu khác:</label>
<textarea id="request" name="request" style="display: block" rows="10"
cols="40"></textarea>
```

Thuộc tính **rows** và **cols** xác định số dòng và chiều rộng. Trong đó, chiều rộng sẽ tương ứng với số ký tự có thể nhập. Tuy nhiên số lượng ký tự trên một dòng chỉ là tương đối vì các ký tự luôn có độ rộng khác nhau. Ví dụ đoạn mã HTML bên trên được hiển thị trên trình duyệt như sau:

Hình 2-29 Vùng nhập văn bản

Có thể xác định kích cỡ của vùng văn bản theo **pixel** bằng cách sử dụng CSS:

```
<label for="request">Các yêu cầu khác:</label>
<textarea  id="request" name="request" style="width:400px;
height:150px;display:block;resize: none"></textarea>
```

Khi sử dụng **textarea**, vùng không gian soạn thảo có thể dễ dàng mở rộng bằng thao tác kéo chuột tại mũi tên dưới cùng góc bên phải của ô nhập. Tính năng này có thể bị vô hiệu hóa bằng cách để thuộc tính **resize** là **none**.

2.11.4 Lựa chọn

f. Radio

Ô nhập với **type** là **radio** giúp tạo ra một ô lựa chọn. Ô lựa chọn này kết hợp với các ô lựa chọn khác tạo thành một nhóm. Người dùng chỉ được chọn 1 tùy chọn trong nhóm này. Ví dụ:

```
<label>Hình thức học:</label>
<input type="radio" name="ca" value="male" id="online" checked><label
for="online">Trực tuyến</label>
<input type="radio" name="ca" value="female" id="offline"><label
for="offline">Tại cơ sở</label>
```

Kết quả:

Hình 2-30 Ô lựa chọn Radio

Lựa chọn này được gọi là **radio button** vì nó tương tự như một thao tác trên các thiết bị đài (radio) thời xưa. Khi đó để chọn kênh trên **radio** người dùng sẽ bấm một nút bấm để chọn kênh và chỉ có thể bấm để kích hoạt 1 kênh tại một thời điểm, khi bấm 1 nút (trạng thái on) thì các nút bấm khác sẽ tự động chuyển sang trạng thái off.

g. Datalist

Phần tử **datalist** xác định một danh sách các tùy chọn được xác định trước cho một phần tử **input**. Ví dụ :

```
<label for="city">Cơ sở học:</label>
<input list="cities" name="city" id="city">
<datalist id="cities">
    <option value="Hà Nội">
    <option value="Thành phố Hồ Chí Minh">
    <option value="Đà Nẵng">
    <option value="Hải Phòng">
    <option value="Cần Thơ">
</datalist>
```

Kết quả:

Hình 2-31 Danh sách lựa chọn

Khi sử dụng **datalist**, người dùng sẽ thấy một danh sách thả xuống để người dùng lựa chọn các giá trị khi nhập liệu. Các phương án lựa chọn được khai báo bằng các phần tử <**option**>. Thuộc tính **list** của phần tử **input** phải tham chiếu đến thuộc tính **id** của phần tử **datalist**. Ngoài việc

lựa chọn các giá trị trong **datalist**, người dùng cũng có thể gõ các giá trị không có trong danh sách.

h. Checkbox

Checkbox được sử dụng để chọn nhiều phương án. Ví dụ:

```
<label>Ca học:</label>
<input type="checkbox" name="ca" value="male" id="sang" checked><label
for="sang">Sáng</label>
<input type="checkbox" name="ca" value="female" id="chieu"><label
for="chieu">Chiều</label>
<input type="checkbox" name="ca" value="other" id="toi"><label
for="toi">Tối</label>
```

Kết quả:

Hình 2-32 Ô lựa chọn checkbox

Trong ví dụ trên, **checkbox** được thiết kế đi cùng với nhãn (**label**) điều này giúp cho người dùng khi sử dụng có thể bấm vào nhãn để lựa chọn phương án chứ không cần bấm vào **checkbox**, như vậy việc lựa chọn sẽ dễ dàng hơn.

i. Select

Phần tử <**select**> định nghĩa một danh sách thả xuống, trong phần tử này định nghĩa các tùy chọn bằng cách khai báo các phần tử **option**. Ví dụ:

```
<select name="cars">
   <option value="volvo">Volvo</option>
   <option value="saab">Saab</option>
   <option value="fiat">Fiat</option>
   <option value="audi">Audi</option>
</select>
```

Để xác định một tùy chọn được chọn sẵn, thêm thuộc tính **selected**:

```
<option value="fiat"selected>Fiat</option>
```

Dùng thuộc tính **size** để xác định những giá trị có thể nhìn thấy, những giá trị còn lại sẽ bị ẩn đi, và người dùng chỉ có thể chọn bằng cách cuộn.

```
<select name="cars" size="3">
   <option value="volvo">Volvo</option>
   <option value="saab">Saab</option>
   <option value="fiat">Fiat</option>
   <option value="audi">Audi</option>
</select>
```

Sử dụng thuộc tính **multiple** để cho phép người dùng chọn nhiều hơn một giá trị:

```
<select name="cars" size="4" multiple>
   <option value="volvo">Volvo</option>
   <option value="saab">Saab</option>
```

```
   <option value="fiat">Fiat</option>
   <option value="audi">Audi</option>
</select>
```

Sử dụng thẻ <**optgroup**> để tạo ra các nhóm lựa chọn:

```
<label for="course-select">Chọn khóa học:</label>
<select id="course-select">
    <optgroup label="Khóa học lập trình">
        <option>C</option>
        <option>Java</option>
        <option>JavaScript</option>
    </optgroup>
    <optgroup label="Cơ sở dữ liệu">
        <option>MySQL</option>
        <option>SQL Server</option>
        <option>MongoDB</option>
    </optgroup>
</select>
```

Khi người dùng bấm để chọn:

Hình 2-33 Các nhóm lựa chọn

Phần tử **<select>** kết hợp với <**option**> sẽ tạo ra một tùy chọn giống cả **radio button** và **checkbox,** khi mà nó có thể tạo ra ô chọn cho phép chỉ chọn 1 phương án duy nhất nhưng cũng cho phép người dùng chọn nhiều phương án (bằng cách khai báo thuộc tính multiple). Phần tử <**select**> cũng có giao diện thuận tiện và được hiển thị khác nhau trên các thiết bị, trình duyệt. Ví dụ trên điện thoại di động chúng thường được hiển thị ở dạng cửa cửa sổ, đôi khi nó cũng cho phép người dùng vuốt để lựa chọn các phương án

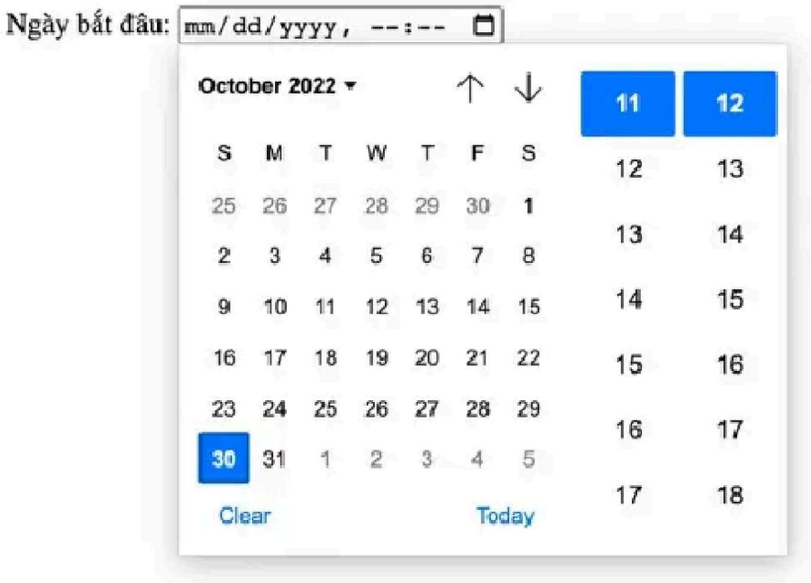
2.11.5 Nhập thời gian

a. Nhập ngày giờ Datetime-local

Ô nhập với type là **datetime-local** quy định một trường đầu vào dạng ngày và giờ, không có múi giờ. Tùy thuộc vào sự hỗ trợ của trình duyệt, một bảng chọn ngày, giờ có thể hiển thị để người dùng thuận tiện trong việc lựa chọn thời gian. Ví dụ:

```
Ngày bắt đầu: <input type="datetime-local" name="start">
```

Kết quả hiện thị như sau:



Hình 2-34 Ô lựa chọn ngày giờ

b. Nhập tháng Month

Phần tử <**input**> với type là **month** cho phép người dùng lựa chọn tháng và năm. Tùy thuộc vào sự hỗ trợ của trình duyệt, một bảng chọn tháng có thể hiển thị, ví dụ:

```
<input type="month" name="start">
```

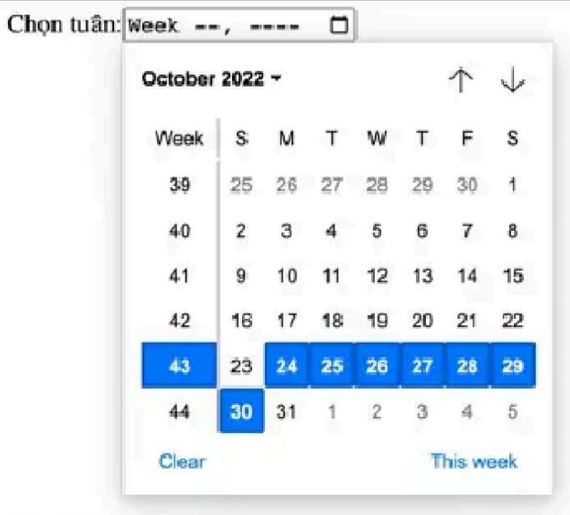
Kết quả:

Hình 2-35 Ô lựa chọn tháng

c. Input Type Week

Phần tử <**input**> với type là **week** cho phép người dùng chọn tuần và năm. Tùy thuộc vào sự hỗ trợ của trình duyệt, một bảng chọn tuần có thể hiển thị, ví dụ:

```
<form>
   Chọn tuần:
   <input type="week" name="week">
</form>
```



Hình 2-36 Ô lựa chọn tuần

d. Input Type Time

Phần tử <**input**> với type là **time** cho phép người dùng chọn một thời gian (không có múi giờ). Tùy thuộc vào sự hỗ trợ của trình duyệt, một bảng chọn thời gian có thể hiển thị trong trường đầu vào.

Ví dụ :

```
Chọn giờ:<input type="time" name="time">
```

Hình 2-37 Ô lựa chọn giờ

2.11.6 Nhập tập tin

Phần tử <**input**> với type là **file** định nghĩa một trường tập tin được chọn và cho tải tập tin lên Server. Ví dụ:

```
Chọn file: <input type="file" name="myFile" accept="image/*,.pdf">
```

Kết quả hiển thị:

Hình 2-38 Ô lựa chọn tập tin

Có thể giới hạn những file cho phép người dùng chọn bằng cách sử dụng thuộc tính **accept** kèm theo các định dạng file. Trường hợp cho phép người dùng chọn nhiều file thì có thể sử dụng thêm khai báo **multiple**

2.11.7 Nút bấm

Phần tử <**input**> với type là button có thể tạo ra nút bấm. Nút bấm có thể bấm bằng cách sử dụng chuột, hoặc sử dụng phím tắt bằng cách khai báo thuộc tính **accesskey.** Ví dụ:

```
<input type="button" value="Add to favorites" accesskey="s">
```

Phần tử button cũng tạo ra một nút bấm:

```
<button type="button" onclick="alert('Hello World!')">
   Click Me!
</button>
```

Phần tử <**button**> có nội dung bên trong, khác với phần tử **input** với type là button là một thẻ trống (không có nội dung). Trong button có thể chứa văn bản, hoặc hình ảnh.

Phần tử nhập với type là reset xác định một nút "thiết lập lại" dùng để xóa toàn bộ các thông tin đã nhập trên biểu mẫu. Phần tử nhập với type là **submit** xác định một nút "xác nhận" để gửi dữ liệu đến máy chủ. Ví dụ:

```
<input type="reset" value="Nhập lại">
<input type="submit" value="Đăng ký">
```

Thuộc tính **action** trong phần tử form xác định hành động sẽ được thực hiện khi **form** được **submit**. Thông thường, dữ liệu của **form** sẽ được gửi đến một trang Web trên máy chủ khi người dùng nhấp vào nút **Submit**. Nếu thuộc tính **action** được bỏ qua, các dữ liệu sẽ được gửi đến chính trang hiện tại. Trong ví dụ ở trên, các dữ liệu của form sẽ được gửi đến một trang trên máy chủ gọi là "/**submit**.**php**". Trang này chứa một server-side script để xử lý các dữ liệu của form.

Thuộc tính **target** của form tương tự phần tử a:

```
<form action="/submit.php" target="_blank">
```

Các thuộc tính **target** xác định nếu kết quả gửi đi sẽ mở trong một tab trình duyệt mới, một frame, hoặc trong cửa sổ hiện hành. Giá trị mặc định là "**_self**", có nghĩa là form được đệ trình trong cửa sổ hiện tại. Để kết quả của form mở trong một tab trình duyệt mới, sử dụng giá trị "**_blank**".

Thuộc tính **method** quy định cụ thể phương thức HTTP (GET hoặc POST) được sử dụng khi gửi dữ liệu của form. Ví dụ sử dụng giao thức GET:

```
<form action="/submit.php" method="get">
```

Hoặc sử dụng giao thức POST:

```
<form action="/submit.php" method="post">
```

Phương thức mặc định khi gửi dữ liệu của form là GET. Khi GET được sử dụng, dữ liệu đã gửi của form sẽ có thể nhìn thấy trong địa chỉ. Chú ý khi dùng GET :

- Gắn thêm form-data vào URL trong tên/cặp giá trị.
- Chiều dài của một URL được giới hạn (2048 ký tự)
- Không bao giờ sử dụng GET để gửi dữ liệu nhạy cảm! (Sẽ được hiển thị trong URL)
- GET phù hợp hơn cho dữ liệu không an toàn, giống như chuỗi truy vấn trong Google.

Giao thức GET sẽ gửi dữ liệu thông qua URL. Ví dụ:

```
/submit.php?name1=value1&name2=value2
```

Phương thức GET có thể sẽ sử dụng tới bộ nhớ đệm (cache), đó là ưu điểm nhưng cũng là 1 vấn đề mà người lập trình thiết kế Web cần chút ý. Ngoài ra, phương thức GET liên quan đến URL vì vậy nó chỉ gửi đi ký tự trong bảng mã ASCII.

Phương thức POST được sử dụng phổ biến hơn khi xác nhận một biểu mẫu. Chú ý:

- POST không có giới hạn kích thước
- Không thể bookmark (đánh dấu)
- Không dùng bộ nhớ đệm

Mỗi trường đầu vào cần phải có một thuộc tính **name**. Nếu thuộc tính **name** bị bỏ qua, dữ liệu đó sẽ không được gửi đi. Ví dụ dưới đây sẽ chỉ **submit** trường đầu vào "Last name":

```
<form action="/action_page.php">
   First name:<br>
   <input type="text" value="Mickey"><br>
   Last name:<br>
   <input type="text" name="lastname" value="Mouse"><br><br>
   <input type="submit" value="Submit">
</form>
```

Chú ý: luôn luôn sử dụng POST nếu dữ liệu dạng chứa thông tin nhạy cảm hoặc cá nhân. Các phương thức POST không hiển thị dữ liệu trong địa chỉ gửi đi. Đặc biệt chú ý khi người dùng bấm **refresh** hoặc nút **back,** dữ liệu có thể sẽ bị gửi đi một lần nữa, trình duyệt sẽ thường cảnh báo người dùng rằng dữ liệu sẽ được gửi đi lần nữa.

2.11.8 Các phần tử biểu mẫu khác

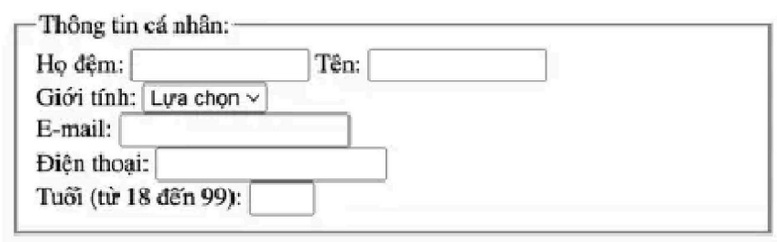
Phần tử <**input**> với **type** là **hidden** sẽ gửi một dữ liệu tới server mà không hiển thị ra cho người dùng biết. Ví dụ muốn gửi một dãy số "**23114306030131**" từ máy người dùng tới server:

```
<input type="hidden" name="userID" value="23114306030131">
```

Thẻ <**fieldset**> được dùng để nhóm dữ liệu liên quan trong một biểu mẫu. Trong khi phần tử <**legend**> định nghĩa một chú thích cho phần tử <**fieldset**>. Ví dụ:

```
<form action="/action_page.php">
   <fieldset>
       <legend>Personal information:</legend>
       First name:<br>
       <input type="text" name="firstname" value="Mickey"><br>
       Last name:<br>
       <input type="text" name="lastname" value="Mouse"><br><br>
       <input type="submit" value="Submit">
   </fieldset>
</form>
```

Mã code trên sẽ được hiển thị trong trình duyệt như sau :



Hình 2-39 Nhóm các phần tử nhập bằng cách sử dụng <fieldset>

Khi nhóm dữ liệu vào một nhóm, thì việc điều khiển nhóm dữ liệu đó sẽ được thực hiện dễ dàng hơn, ví dụ có thể vô hiệu (**disabled**) hoặc hiển thị (**display**) cả một nhóm dữ liệu thay vì vô hiệu, hoặc hiển thị cho từng điều khiển.

Phần tử **<label>** (nhãn) được sử dụng để khai báo cùng với các phân tử nhập:

- Các phần tử có thể sử dụng nhãn bao gồm **input, meter, progress, select, textarea**
- Nhãn sẽ được **đọc to** khi người dùng sử dụng trình đọc màn hình và bấm chọn vào điều khiển
- Nhãn giúp người dùng dùng bấm trên các vùng nhỏ như **radio button** hay **checkbox** dễ dàng hơn.

Phần tử <**label**> sẽ có thuộc tính for, thuộc tính này phải giống như thuộc tính id của phần tử được gán nhãn. Ngoài ra một phần tử được gán nhãn cũng có thể đặt bên trong phần tử **label**. Ví dụ gán nhãn cho **radio button**:

```
<form action="/action_page.php">
 <input type="radio" id="html" name="fav_language" value="HTML">
 <label for="html">HTML</label><br>
```

```
 <input type="radio" id="css" name="fav_language" value="CSS">
 <label for="css">CSS</label><br>
 <input type="radio" id="JavaScript " name="fav_language" value="JavaScript ">
 <label for="JavaScript ">JavaScript </label><br><br>
 <input type="submit" value="Submit">
</form>
```

Phần tử **<progress>** không phải phần tử nhập, nó được sử dụng để thể hiện tiến độ hoàn thành một tác vụ nào đó, ví dụ:

```
<label for="file">Downloading progress:</label>
<progress id="file" value="32" max="100"> 32% </progress>
```

Kết quả hiển thị:

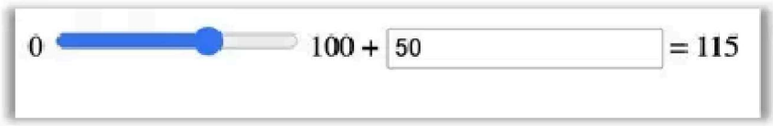
Hình 2-40 Phần tử hiển thị tiến độ <process>

Phần tử <**process**> thường được hoạt động cùng với mã JavaScript để cập nhật tiến độ. Bên cạnh phần tử <**process**>, phần tử <**meter**> cũng hiển thị tương tự. Tuy nhiên mục đích của 2 phần tử này khai báo 2 nội dung khác nhau. Trong khi <**process**> được sử dụng để khai báo tiến độ, còn <**metter**> được sử dụng để thể hiện một đại lượng đo lường (ví dụ như tốc độ, dung lượng ...)

Phần tử <**output**> được sử dụng để hiển thị kết quả đầu ra. Ví dụ thực hiện tính toán và hiển thị kết quả trong một phần tử **output**

```
<form oninput="x.value=parseInt(a.value)+parseInt(b.value)">
   0 <input type="range" id="a" name="a" value="50"> 100 +
   <input type="number" id="b" name="b" value="50"> =
   <output name="x"></output>
</form>
```

Kết quả:



Hình 2-41 Hiển thị kết quả đầu ra bằng <output>

Trong ví dụ trên, phần tử <**form**> có vai trò trung gian lấy dữ liệu sẽ có thuộc tính **oninput** và hàm JavaScript tương ứng để tính toán, trả về dữ liệu để hiển thị trên phần tử <**output**>

2.11.9 Các thuộc tính áp dụng cho phần tử nhập

Có thể thấy rằng các phần tử nhập có rất nhiều thuộc tính. Phần dưới đây sẽ tóm tắt cách sử dụng các thuộc tính này.

Thuộc tính **value** xác định giá trị ban đầu, đó cũng là giá trị sẽ gửi đi khi người dùng không nhập gì và điều khiển đó:

```
First name:<br>
<input type="text" name="firstname" value="John">
```

Thuộc tính **readonly** xác định rằng trường đầu vào là chỉ đọc (không thể thay đổi):

```
First name:<br>
<input type="text" name="firstname" value="John" readonly>
```

Thuộc tính **disable** xác định rằng trường đầu vào bị vô hiệu hóa, dữ liệu vô hiệu hóa cũng sẽ không được gửi đi khi submit form.

```
First name:<br>
<input type="text" name="firstname" value="John" disabled>
```

Thuộc tính **size** xác định kích thước cho trường đầu vào:

```
First name:<br>
<input type="text" name="firstname" value="John" size="40">
```

Thuộc tính **maxlength** xác định chiều dài tối đa cho phép, chú ý khi vượt quá ký tự cho phép trình duyệt không cho gõ tiếp chứ không đưa ra thông báo gì:

```
First name:<br>
<input type="text" name="firstname" maxlength="10">
```

Thuộc tính **autocomplete** xác định liệu một form hay một trường đầu vào có tự động điền biểu mẫu hay không:

```
E-mail: <input type="email" name="email" autocomplete="off">
```

Khi **autocomplete** bật, trình duyệt tự động hoàn thành các giá trị đầu vào dựa trên các giá trị mà người dùng đã nhập. Thuộc tính **autocomplete** làm việc với <**form**> và kiểu <**input**> với thuộc tính type là **text**, **search**, **url**, **tel**, **email**, **password**, **datepicker**, **range** và **color**. Có thể đặt **autocomplete** "on" đối với form, và "off" cho các trường đầu vào cụ thể, hoặc ngược lại. Trong một số trình duyệt, người dùng có thể cần kích hoạt chức năng **autocomplete** để hoàn thành việc này.

Thuộc tính **novalidate** là một thuộc tính của <form> để xác định rằng các dữ liệu form không cần được được kiểm tra khi submit:

```
<form action="/action_page.php" novalidate>
   E-mail: <input type="email" name="user_email">
   <input type="submit">
</form>
```

Thuộc tính **autofocus** xác định rằng trường đầu vào sẽ tự động được **focus**, ví dụ

```
First name:<input type="text" name="fname" autofocus>
```

Thuộc tính **height** và **width** xác định chiều cao và chiều rộng của một phần tử:

```
<input type="image" src="img_submit.gif" alt="Submit" width="48" height="48">
```

Thuộc tính **list** chỉ đến một phần tử <**datalist**> chứa các lựa chọn được xác định trước cho một phần tử <**input**>.

```
<input list="browsers">
<datalist id="browsers">
   <option value="Internet Explorer">
   <option value="Firefox">
   <option value="Chrome">
   <option value="Opera">
   <option value="Safari">
</datalist>
```

Thuộc tính **min** và **max** xác định giá trị tối thiểu và tối đa cho một phần tử input. Thuộc tính **min** và **max** làm việc với các loại kiểu: **number**, **range**, **date**, **datetime-local**, **month**, **time** và **week**.

```
Nhập một ngày trước năm 1980
<input type="date" name="bday" max="1979-12-31">

Nhập một ngày sau năm 2000
<input type="date" name="bday" min="2000-01-02">

Nhập một số có giá trị trong khoảng từ 1 đến 5
<input type="number" name="quantity" min="1" max="5">
```

Thuộc tính **multiple** xác định rằng người dùng được phép nhập nhiều hơn một giá trị trong phần tử <**input**>. Thuộc tính **multiple** làm việc với các loại đầu vào như email, và file.

```
Lựa chọn ảnh: <input type="file" name="img" multiple>
```

Thuộc tính **pattern** chỉ định một biểu thức nguyên mẫu mà các giá trị của phần tử **input** được kiểm tra đối chiếu. Thuộc tính **pattern** làm việc với các loại đầu vào: **text**, **search**, **url**, **tel**, **email**, và **password**. Sử dụng thuộc tính **title** để mô tả giúp người dùng hình dùng được nên nhập thế nào:

```
Mã quốc gia: <input type="text" name="country_code" pattern="[A-Za-z]{3}"
title="mã quốc gia">
```

Thuộc tính **placeholder** được sử dụng để hiển thị các gợi ý trong các ô nhập:

```
Tên: <input type="text" name="fname" placeholder="First name">
```

Thuộc tính **required** xác định rằng trường đầu vào phải được điền đầy đủ:

```
Tên: <input type="text" name="username" required>
```

Thuộc tính **step** xác định khoảng số nhảy cho phần tử nhập ở dạng số. Ví dụ: nếu step= “3”, những số hợp lệ có thể là -3,0,3,6… Ví dụ:

```
<input type="number" name="points" step="3">
```

2.12 NỘI DUNG ĐA PHƯƠNG TIỆN

2.12.1 Hình ảnh Bitmap

Hình ảnh **bitmap** là các hình ảnh lưu trữ dữ liệu theo dạng điểm ảnh. Các hình ảnh này có thể được đưa vào tài liệu HTML bằng phần tử <**img**>. Các phần tử <**img**> chỉ chứa các thuộc tính, và không có thẻ đóng. Các thuộc tính **src** xác định URL (địa chỉ Web) của hình ảnh, ví dụ:

```
<img src="url">
```

Các ảnh có thể được tải ở trong thư mục khác với thư mục hiện tại. Nếu không được chỉ định, trình duyệt sẽ tìm thấy hình ảnh trong cùng thư mục với trang Web. Thông thường để lưu trữ hình ảnh trong một thư mục phụ. Sau đó, phải bao gồm tên thư mục trong thuộc tính src:

```
<img src="/images/html5.gif" alt="HTML5 Icon" style="width:128px;height:128px;">
```

Một số trang Web lưu trữ hình ảnh của họ trên các máy chủ khác hoặc sử dụng một hình ảnh có sẵn từ các trang Web khác:

```
<img src="https://www.w3schools.com/images/w3schools_green.jpg" alt="W3Schools.c
om">
```

Thuộc tính **alt** cung cấp một văn bản thay thế cho hình ảnh, nếu vì một lý do nào đó không thể xem được ảnh (vì kết nối chậm, một lỗi trong các thuộc tính **src**, hoặc nếu người dùng sử dụng một trình đọc màn hình) thuộc tính **alt** sẽ được sử dụng.

```
<img src="img_chania.jpg" alt="Flowers in Chania">
```

Lưu ý: Thuộc tính **alt** là bắt buộc. Một trang Web sẽ không thể xác thực là "chuẩn" nếu các thẻ <**img**> thiếu thuộc tính này.

Sử dụng các thuộc tính **width** và **height** để xác định chiều rộng và chiều cao của ảnh.

```
<img src="img_girl.jpg" alt="Girl in a jacket" width="500" height="600">
```

Ngoài ra cũng thể sử dụng thuộc tính **style** để xác chỉ định chiều rộng và chiều cao như sau:

```
<img src="img_girl.jpg" alt="Girl" style="width:500px;height:600px;">
```

Các thuộc tính **width** và **height** luôn sử dụng đơn vị là **pixel**, nếu chiều rộng và chiều cao không được xác định trước, trang Web có hiệu ứng tải "nhấp nháy" khiến cho người dùng cảm thấy khó chịu.

Để sử dụng hình ảnh làm liên kết, đặt <**img**> thẻ bên trong <**a**> thẻ:

```
<a href="tutorial.html">
<img src="smiley.gif" alt="Hướng dẫn
HTML" style="width:42px;height:42px;border:0;">
</a>
```

Chú ý một hình ảnh có thể trở thành một liên kết để người dùng nhấp chọn. Một hình ảnh phức tạp cũng có thể đưa nhiều liên kết vào các vị trí khác nhau bằng cách khai báo một phần tử <**map**> **.** Ví dụ:

```
<img src="workplace.jpg" alt="Workplace" usemap="#workmap" width="400"
height="379">
<map name="workmap">
  <area shape="rect" coords="34,44,270,350" alt="Computer" href="computer.htm">
  <area shape="rect" coords="290,172,333,250" alt="Phone" href="phone.htm">
  <area shape="circle" coords="337,300,44" alt="Cup of coffee"
href="coffee.htm">
</map>
```

Đoạn mã này sử dụng thẻ <**map**> để khai báo 3 vùng, bao gồm 2 vùng là hình chữ nhật (chiếc máy tính và điện thoại) và một vùng hình tròn (tách cafe). Người đọc nội dung trên trình duyệt sẽ có thể nhấp chuột vào ba vùng này để được chuyển đến các trang Web khác nhau.

Hình 2-42 Kết quả hiển thị một hình ảnh

2.12.2 Hình ảnh vector

Ảnh **SVG** (**S**calable **V**ector **G**raphics) được sử dụng để thể hiện hình ảnh đồ họa vector 2 chiều và thường được sử dụng khác phổ biến ngày này. Hình ảnh **vector** có ưu điểm là nó có thể dễ dàng thu phóng một cách thoải mái và không làm giảm chất lượng bởi nó là định dạng ảnh vector. Ảnh SVG được tạo ra từ mã lệnh hoặc các phần mềm vector như Adobe Illustrator, CorelDRAW. SVG cũng sử dụng cú pháp của **xml** và được khai báo trong tài liệu HTML bằng thẻ <**svg**> . Các hình ảnh **vector** này có một số phần tử để tạo ra các đường, đa giác, vòng tròn, văn bản và hình ảnh đồ họa. Ví dụ:

Phần tử <**circle**> tạo ra hình tròn:

```
<svg width="100" height="100">
   <circle cx="50" cy="50" r="40" stroke="green" stroke-width="4" fill="yellow"
/>
</svg>
```

Phần tử <**rect**> tạo ra hình chữ nhật

```
<svg width="400" height="100">
   <rect width="400" height="100" style="fill:rgb(0,0,255);stroke-
width:10;stroke:rgb(0,0,0)" />
```

```
</svg>
```

Thuộc tính **rx ry** giúp tạo ra hình chữ nhật bo tròn.

```
<svg width="400" height="180">
   <rect x="50" y="20" rx="20" ry="20" width="150" height="150"
         style="fill:red;stroke:black;stroke-width:5;opacity:1" />
</svg>
```

Phần tử <**polygon**> tạo ra các hình đa giác bằng việc cung cấp các tọa độ điểm.

```
<svg width="300" height="200">
   <polygon points="100,10 10,78 40,198 160,198 190,78"
            style="fill:lime;stroke:purple;stroke-width:5;fill-rule:evenodd;" />
</svg>
```

Phần tử <**text**> để tạo tạo ra chữ, phần tử <**ellipse**> tạo ra hình ellipse. Các phần tử này được đặt cùng nhau trong một phần tử svg để tạo ra một hình ảnh. Ví dụ:

```
<svg height="130" width="500">
   <ellipse cx="100" cy="70" rx="85" ry="55" fill="#123456" />
   <text fill="#ffffff" font-size="45" font-family="Verdana" x="50"
y="86">SVG</text>
</svg>
```

Kết quả đoạn mã trên, một hình ảnh được xuất ra:

Hình 2-43 Hình ảnh vector được tạo ra

Đồ họa **vector** rất quan trọng trong thiết kế các trang Web hiện đại, thông thường các biểu tượng, logo của các Website thường được thiết kế ở dạng **vector** và xuất ra định dạng file **svg** hoặc file mã từ đó dễ dàng nhúng vào các Website. Đồ họa **vector** thường có thiết kế hiện đại, dung lượng nhỏ, không bị vỡ khi phóng to. Chú ý, cần phân biệt giữa **SVG** và **Canvas**: **SVG** là ngôn ngữ để mô tả đồ họa bằng XML trong khi **Canvas** vẽ đồ họa bằng JavaScript. Các hàm vẽ Javascript sẽ vẽ trên một phần tử <**canvas**>

2.12.3 Âm thanh và phim

Một Website có thể bổ sung các nội dung âm thanh để tăng thêm sự sinh động. Các file âm thanh có thể được tạo ra kèm theo phần điều khiển bằng thẻ <**audio**>. Phần tử <**source**> bên trong cũng được sử dụng để xác định các nguồn nội dung sẽ được trình chiếu. Ví dụ:

```
<audio controls>
   <source src="horse.mp3" type="audio/mpeg">
   Your browser does not support the audio element.
</audio>
```

Một đoạn phim có thể đưa vào tài liệu HTML bằng thẻ <**video**>. Phần tử <**source**> bên trong cũng được sử dụng để xác định các nguồn nội dung sẽ được trình chiếu. Ví dụ:

```
<video width="320" height="240" controls>
  <source src="movie.mp4" type="video/mp4">
  <source src="movie.ogg" type="video/ogg">
  Your browser does not support the video tag.
</video>
```

Có thể đặt bất kỳ loại file đa phương tiện nào trong thuộc tính src của phần tử <**video**> như Flash movies (.swf), AVI's (.avi), và MOV's (.mov) trong thẻ videos.

- .swf files - là các file được tạo bởi chương trình Macromedia's Flash.
- .wmv files - được tạo bởi Microsoft's Window's Media Video.
- .mov files - được tạo bởi định dạng Apple's Quick Time Movie.
- .mpeg files - được tạo bởi Moving Pictures Expert Group.

Một số thuộc tính khác được sử dụng để khai báo thêm các thuộc tính điều khiển nội dung đa phương tiện này:

Bảng 2-3 Bảng thuộc tính hỗ trợ cho phần tử audio/video

Thuộc tính	Mô tả
autoplay	Thuộc tính này dùng để kích hoạt/ tắt tính năng tự khởi động
loop	xác định nội dung sẽ được phát đi phát lại (truyền vào giá trị), hoặc chạy duy nhất 1 lần (đặt là false).
controls	Thiết lập hiển thị/tắt chức năng điều khiển video
width/height	Độ rộng/chiều cao của đối tượng thể hiện bằng pixel.
src	Địa chỉ URL của đối tượng được nhúng.
preload	Thiết lập video sẽ được tải trước khi người dùng bấm chạy
muted	Tắt/Bật tiếng của video

Với những video và âm thanh có phụ đề, có thể khai báo file này thông qua phần tử <**track**>. Ví dụ một đoạn video có phụ đề kèm theo:

```
<video controls
       src="/media/cc0-videos/friday.mp4">
    <track default
           kind="captions"
           srclang="en"
```

```
            src="/media/examples/friday.vtt">
    Download the
    <a href="/media/cc0-videos/friday.mp4">MP4</a>
    video, and
    <a href="/media/examples/friday.vtt">subtitles</a>.
</video>
```

Kết quả hiển thị:

Hình 2-44 Một video với phụ đề kèm theo

Với phần tử video có thể khai báo thêm thuộc tính **poster** để xác định hình ảnh sẽ hiển thị khi video chưa được trình chiếu.

Tài liệu HTML cũng có thể phát các video hoặc âm thanh từ các nguồn trên mạng và sử dụng chính các trang đó để phát. Đây là kỹ thuật sử dụng phần tử <**iframe**> để nhúng các Website vào tài liệu HTML của mình. Ví dụ để phát một đoạn video trên Youtube bằng phần tử <**iframe**>:

```
<iframe width="560" height="315" src="https://www.youtube.com/embed/pQN-pnXPaVg"
title="YouTube video player" frameborder="0" allow="accelerometer; autoplay;
clipboard-write; encrypted-media; gyroscope; picture-in-picture"
allowfullscreen></iframe>
```

2.12.4 Tập tin PDF

Nhúng một tài liệu PDF trong một tài liệu HTML bằng cách sử dụng thẻ <**embed**> như sau:

```
<embed src="https://minh-blogs.Web.app/files/article.pdf" width="500"
height="375" type="application/pdf">
```

Chú ý phần tử <**embed**> cũng được sử dụng để nhúng các tài liệu **html**, file **video**, file **âm thanh** và cả những ứng dụng khác như **applet**, tuy nhiên điều này không còn được sử dụng phổ biến do có nhiều hạn chế. Để hiển thị hình ảnh nên sử dụng phần tử <**img**>, để hiển thị video và âm thanh sử dụng phần tử <**video**> và <**audio**> trong khi hiển thị file html thì nên sử dụng phần tử <**iframe**>

THỰC HÀNH

CÂU HỎI ÔN TẬP LÝ THUYẾT

Hãy chọn phương án đúng nhất

1. Thẻ nào để tạo ra chữ đậm?
A. string
B. b
C. h1
D. head

2. Thành phần html nào tạo ra một ngắt dòng?
A. br
B. break
C. hr
D. p

3. Thẻ nào định nghĩa một văn bản quan trọng?
A. i
B. important
C. strong
D. em

4. Thẻ nào được sử dụng để nhấn mạnh văn bản?
A. p
B. i
C. strong
D. em

5. Thuộc tính nào được sử dụng để hợp 2 hàng trong 1 bảng?
A. rowpadding
B. cellpadding
C. rowspan
D. colspan

6. Để tạo liên kết trong thẻ a sử dụng thuộc tính gì?

A. href
B. src
C. link
D. ref

7. Đơn vị nào được sử dụng cho thuộc tính chiều cao, rộng của ảnh?

A. Pixels
B. Centimeters
C. Dot
D. Inches

8. Tại sao lại sử dụng các phần tử cấu trúc?

A. Vì nó có nhiều thuộc tính hữu ích
B. Vì nó tạo ra các giao diện phù hợp với Website
C. Vì các phần tử khác không giải quyết được yêu cầu
D. Tạo ra các thành phần ngữ nghĩa

9. Thẻ nào được tạo ra cách gạch đầu dòng?

A. ul
B. ol
C. bullet
D. list

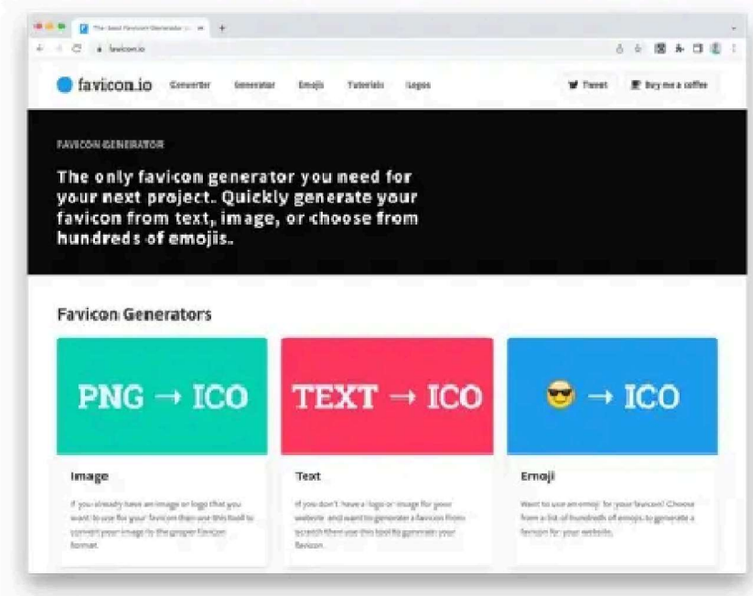
10. Đoạn mã HTML nào là ĐÚNG để tạo một hyperlink?

A. W3Schools.com
B. W3Schools
C. W3Schools.com
D. <a>http://www.w3schools.com

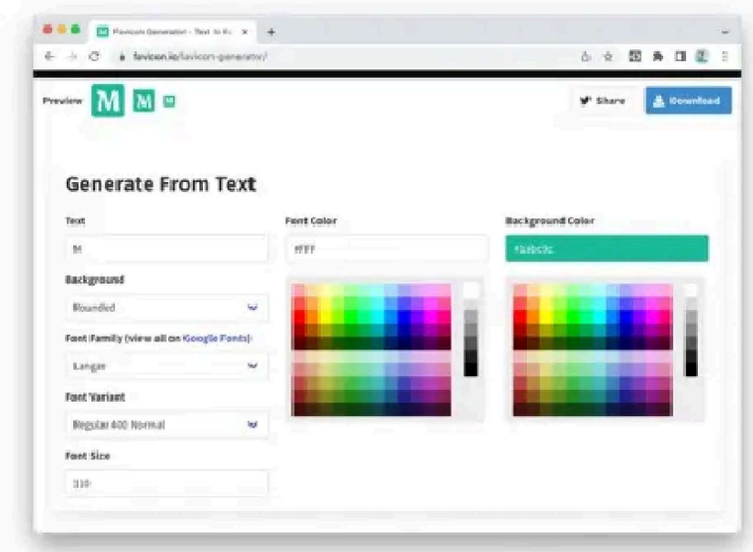
BÀI TẬP TỰ THỰC HÀNH

Bài 1. Tạo favicon

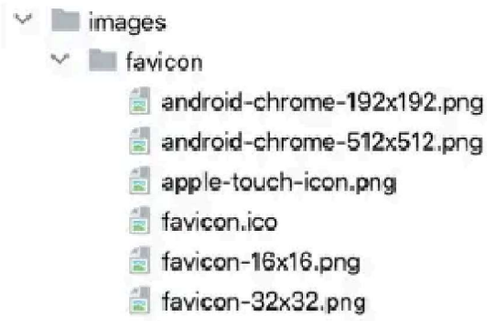
Vào trang https://favicon.io/ sau đó chọn Text -> Ico để chuyển một chữ cái thành ảnh favicon



Chọn một chữ cái phù hợp, (ví dụ chữ M như hình dưới), lựa chọn màu sắc chữ màu nền phù hợp để tạo ảnh favicon.



Tải toàn bộ ảnh về đặt trong thư mục images/favicon



Sử dụng đoạn mã sau để khai báo trong phần head

```
<link rel="apple-touch-icon" sizes="180x180" href="images/favicon/apple-touch-
icon.png">
<link rel="icon" type="image/png" sizes="32x32" href="images/favicon/favicon-
32x32.png">
<link rel="icon" type="image/png" sizes="16x16" href="images/favicon/favicon-
16x16.png">
<link rel="manifest" href="/Webmanifest.json">
```

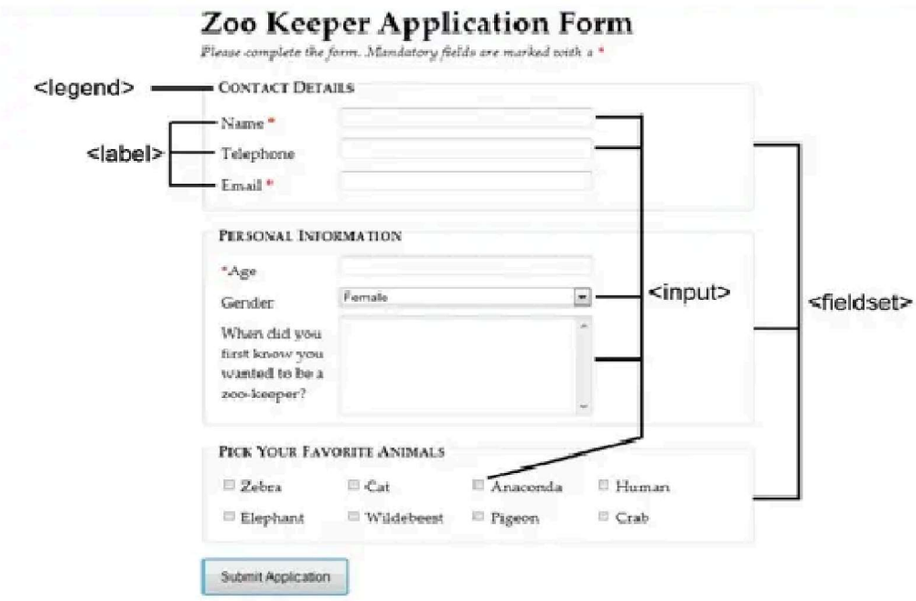
Tạo ra file Webmanifest.json với nội dung như sau:

```
{
  "name": "",
  "short_name": "",
  "icons": [
    {
      "src": "images/favicon/android-chrome-192x192.png",
      "sizes": "192x192",
      "type": "image/png"
    },
    {
      "src": "images/favicon/android-chrome-512x512.png",
      "sizes": "512x512",
      "type": "image/png"
    }
  ],
  "theme_color": "#ffffff",
  "background_color": "#ffffff",
  "display": "standalone"
}
```

Bài 2. Hãy tạo một giao diện như sau với các ảnh là svg

Các ảnh có thể là tự thiết kế ảnh hoặc chọn ảnh trên mạng

Bài 3. Tạo form



TÀI LIỆU THAM KHẢO

[1] Anne Boehm, Zak Ruvalcaba (2018) Murach’s HTML5 and CSS3, Murack.

[2] Responsive Web Design with HTML5 and CSS: Build future-proof responsive Websites using the latest HTML5 and CSS techniques, 3rd, (2020), Ben Frain, Packt Publishing

[3] Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics (2018), Jennifer Robbins, O'Reilly Media

CHƯƠNG 3: CÁC THUỘC TÍNH ĐỊNH DẠNG CƠ BẢN CỦA CSS

Chương này sẽ giới thiệu về ngôn ngữ CSS - một ngôn ngữ được dùng để định dạng các phần tử do ngôn ngữ đánh dấu văn bản tạo ra. Người học sẽ được học các thuộc tính cơ bản trong ngôn ngữ CSS. Bên cạnh đó, cách nhúng CSS nhằm kiểm soát bố cục và tạo ra phong cách thống nhất cho một trang Web cũng sẽ được trình bày.

3.1 GIỚI THIỆU

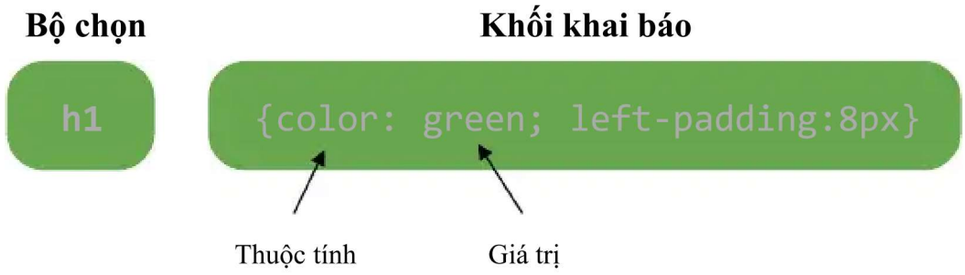
3.1.1 CSS là gì?

CSS là viết tắt của Cascading Style Sheets được sử dụng để mô tả cách các phần tử HTML được hiển thị. CSS có thể kiểm soát bố cục của các trang Web bằng việc khai báo các style (định dạng) nhờ đó mà nó tiết kiệm được công sức định dạng các phần tử HTML.

3.1.2 Cú pháp CSS

Cú pháp của CSS bao gồm **Bộ chọn** và **Khối khai báo**:

- **Bộ chọn** (Selector): trỏ đến phần tử HTML mà muốn tạo kiểu
- **Khối khai báo** (Declaration): chứa một hoặc nhiều cặp "thuộc tính", "giá trị"



Hình 3-1 Cú pháp của một khai báo CSS

Ví dụ:

```
p {
 color: red;
 text-align: center;
}
```

Trong ví dụ trên **p** là Selector để chọn ra các phần tử <**p**>, khối khai báo bắt đầu từ dấu ngoặc nhọn chỉ ra phần tử <**p**> này có chữ bên trong màu đỏ và căn giữa. Một khai báo CSS luôn kết thúc bằng dấu chấm phẩy (;) và các khối khai báo được bao quanh bởi các dấu ngoặc nhọn {}.

3.1.3 Ghi chú thích cho CSS

Các chú thích (Comment) trong CSS:

- Được sử dụng để giải thích mã hoặc chỉnh sửa mã
- Bị bỏ qua bởi các trình duyệt.

- Bắt đầu bằng /* và kết thúc bằng */
- Có thể trải dài trên nhiều dòng

Ví dụ:

```
p {
 color: red;
 /* This is a single-line comment */
 text-align: center;
}

/* This is a
multi-line comment */
```

3.2 CHÈN CSS VÀO TÀI LIỆU HTML

Ba cách để chèn CSS vào tài liệu HTML

- CSS bên ngoài (External)
- CSS nội bộ (Internal)
- CSS nội tuyến (Inline)

3.2.1 CSS bên ngoài

CSS bên ngoài được lưu trữ trong 1 file có phần mở rộng **.css**. Trang HTML phải tham chiếu đến tệp **.css** bên ngoài bằng phần tử **<link>**, phần tử này được đặt trong phần tử **<head>.** Thay đổi giao diện của trang Web bằng cách thay đổi file **.css.** CSS bên ngoài có thể được áp dụng cho nhiều file HTML.

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
<header>
    <h1>Minh's Page</h1>
    <p>Lập kế hoạch - Ghi chú thích - <b>Viết mã lệnh</b> - Kiểm thử</p>
</header>
</body>
</html>
```

Ví dụ nội dung của file **style.css**

```
body {
    font-family: Arial;
}
header {
    background: #1abc9c;
    color: white;
    padding: 60px;
    text-align: center;
}
```

Các khai báo css trên sẽ áp dụng cho các nội dung trong phần tử <body> sử dụng font chữ Arial. Và phần tử <header> có màu nền #1abc9c, màu chữ trắng, padding (phần khoảng trống trắng) là 60px và chữ căn giữa. Kết quả khi hiển thị trên trình duyệt thì tài liệu HTML trên sẽ hiển thị như sau:

Hình 3-2 Kết quả sau khi áp dụng css bên ngoài

3.2.2 CSS nội bộ

CSS nội bộ được khai báo trong tập tin HTML. Chính vì vậy nó chỉ được sử dụng bởi 1 tập tin HTML duy nhất. CSS nội bộ được khai báo trong phần tử <**style**>

Ví dụ những khai báo thuộc tính CSS đặt trong file style.css ngoài như trên có thể đặt trong thẻ <**style**> trong tài liệu HTML như sau:

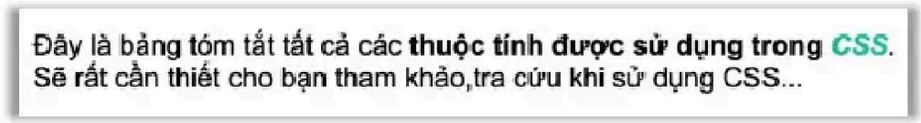
```
<!DOCTYPE html>
<html>
<head>
<style>
body {
    font-family: Arial;
}
header {
    background: #1abc9c;
    color: white;
    padding: 60px;
    text-align: center;
}
</style>
</head>
<body>
<header>
    <h1>Minh's Page</h1>
    <p>Lập kế hoạch - Ghi chú thích - <b>Viết mã lệnh</b> - Kiểm thử</p>
</header>
</body>
</html>
```

3.2.3 CSS nội tuyến

CSS nội tuyến có thể được sử dụng để áp dụng kiểu cho **một** phần tử. Để sử dụng css nội tuyến, thêm thuộc tính **style** trực tiếp vào phần tử HTML. Ví dụ:

```
Đây là bảng tóm tắt tất cả các <span style="font-weight: bold">thuộc tính được
sử dụng trong <em style="color: #1abc9c">CSS</em></span>. Sẽ rất cần thiết cho
bạn tham khảo,tra cứu khi sử dụng CSS...
```

Kết quả khi hiển thị trên trình duyệt:



Hình 3-3 Kết quả khi áp dụng CSS nội tuyến

Chú ý, nên hạn chế sử dụng CSS nội tuyến vì những cấu hình thuộc tính này sẽ không thể sử dụng được cho các phần tử HTML khác. Việc chỉnh sửa thiết kế sau này cũng rất mất thời gian công sức.

3.2.4 Kết hợp CSS

- Trong tài liệu HTML có thể sử dụng nhiều loại CSS.
- Giá trị từ CSS đọc sau cùng sẽ được sử dụng và CSS đọc trước sẽ bị ghi đè (mất tác dụng)

Giả sử rằng một CSS ngoài có kiểu sau cho phần tử <**h1**>:

```
h1 {
 color: navy;
}
```

Sau đó, giả sử rằng một CSS nội bộ cũng có kiểu sau cho phần tử <**h1**>:

```
h1 {
 color: orange;
}
```

Nếu CSS ngoài được khai báo trước, nó sẽ **không có tác dụng** vì bị **CSS nội bộ** ghi đè.

```
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
<style>
h1 {
 color: orange;
}
</style>
</head>
```

Ngược lại nếu **CSS ngoài** khai báo sau, nó sẽ ghi đè lên CSS nội bộ:

```
<head>
<style>
h1 {
 color: orange;
}
</style>
```

```
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
```

3.2.5 Khai báo !important

Khai báo **!important** khiến cho việc khai báo một thuộc tính kiểu trở lên **quan trọng,** nó sẽ có quyền ưu tiên cao và không để những khai báo sau ghi đè lên. Ví dụ:

```
#myid {
    background-color: blue !important;
}
p {
    background-color: red;
}
.myclass {
    background-color: gray;
    padding: 20px;
    color: white;
}
```

Trường hợp này nếu có 1 thẻ p có id là myid, có class là myclass thì sẽ có màu nền là blue. Ví dụ:

```
<p id="myid" class="myclass">Nội dung thẻ p</p>
```

Kết quả hiển thị:

Chú ý, nên hạn chế việc sử dụng **!important**, chỉ dùng nó khi không thể thay đổi thứ tự của các CSS. Ví dụ các khai báo css đặt ở trong các file css khác nhau. Như trường hợp trên việc sử dụng **!important** là không cần thiết, vì có thể đưa khối khai báo thuộc tính **#myid** xuống dưới và không cần sử dụng tới **!important** nữa.

3.2.6 Thuộc tính all

Khi một phần tử thuộc một phần tử, ta gọi đó là phần tử con và nó sẽ được áp dụng các thuộc tính của phần tử cha. Tuy nhiên có thể sử dụng thuộc tính **all** đặt lại những giá trị thuộc tính mà phần tử, quyết định nó có sử dụng hoặc không sử dụng thuộc tính của phần tử cha. Các giá trị có thể thiết lập cho thuộc tính all:

- **initial:** Thiết lập toàn bộ các thuộc tính trở về giá trị mặc định
- **inherit:** Thiết lập toàn bộ các thuộc tính kế thừa từ phần tử cha
- **unset** (mặc định)**:** Thiết lập toàn bộ các thuộc tính kế thừa từ phần tử cha nếu đó là những thuộc tính mặc định sẽ kế thừa từ cha.

Ví dụ sau đây, sẽ thiết lập các phần tử <**div**> đặt trong phần tử <**body**> sẽ không kế thừa từ phần tử cha. Như vậy cho dù phần tử <**div**> có đặt trong phần tử <**body**> nó vẫn không có màu chữ là đỏ.

```
body {
  font-size: small;
  color: red;
}

div {
  all: initial;
}
```

3.3 CÁC LOẠI BỘ CHỌN

3.3.1 Bộ chọn CSS

Bộ chọn (Selector) CSS được sử dụng để "tìm" (hoặc chọn) các phần tử HTML mà muốn tạo kiểu, có thể chia thành **5 loại Bộ chọn chính**:

- Bộ chọn **đơn giản** (Simple): chọn thành phần dựa trên tên thẻ, id, lớp
- Bộ chọn **kết hợp** (Combinator): chọn các yếu tố dựa trên mối quan hệ
- Bộ chọn **lớp giả** (Pseudo-class): chọn các thành phần dựa trên trạng thái
- Bộ chọn **phần tử giả** (Pseudo-elements): chọn và định kiểu một phần của phần tử
- Bộ chọn **thuộc tính** (Attribute Selectors): chọn thành phần dựa trên thuộc tính

3.3.2 Bộ chọn đơn giản

Bộ chọn đơn giản sẽ được tìm hiểu trước, đây là bộ chọn được sử dụng rất phổ biến, nó bao gồm các loại sau:

1) Bộ chọn **phần tử**
2) Bộ chọn theo **id**
3) Bộ chọn theo **class**
4) Bộ chọn ***

Bộ chọn phần tử chọn các phần tử HTML dựa trên tên phần tử. Ví dụ chọn toàn bộ các phần tử <**body**>:

```
body {
    font-family: Arial;
    margin: 0;
}
```

Bộ chọn theo id sử dụng thuộc tính id của một phần tử HTML để chọn một phần tử cụ thể. Id của một phần tử là duy nhất trong một tài liệu HTML, vì vậy bộ chọn id được sử dụng để chọn một phần tử duy nhất. Trong CSS, để chọn một phần tử với id cụ thể, viết # kèm theo id của phần tử. Chú ý, khi đặt tên id không được bắt đầu bởi số, không chứa khoảng trắng. Ví dụ dưới đây sẽ được áp dụng cho phần tử HTML với id = "avatar":

```
<style>
    #avatar {
        border-radius: 50%;
    }
```

```
</style>
<h1>Phạm Dương Minh</h1>
<img id="avatar" src="images/me.jpg" height="200px"/>
```

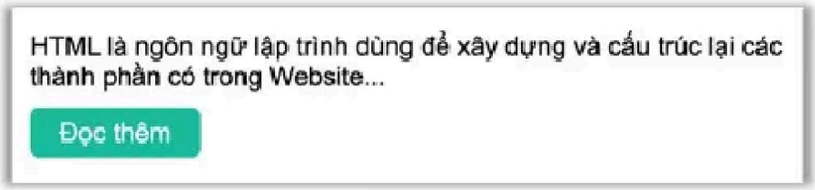
Kết quả hiển thị:

Hình 3-4 Bức ảnh được định danh bằng id áp dụng CSS

Bộ chọn theo class chọn các phần tử HTML với thuộc tính lớp cụ thể. Để chọn các phần tử với một class cụ thể, sử dụng dấu chấm (.) trước tên class. Như Trong ví dụ dưới đây, tất cả các phần tử HTML có class = "center" sẽ có màu đỏ và căn giữa:

```
<style>
.readmore {
    align-self: flex-end;
    background-color: #1abc9c !important;
    border-radius: 5px;
    color: white;
    padding: 6px 18px;
    text-decoration: none;
}
</style>
<p>HTML là ngôn ngữ lập trình dùng để xây dựng và cấu trúc lại các thành phần có
trong Website...</p><a class="readmore">Đọc thêm</a>
```

Kết quả khi hiển thị trên trình duyệt:



Hình 3-5 Áp dụng CSS trên các phần tử có class là readmore

Cũng có thể chỉ định rằng chỉ các phần tử HTML cụ thể mới bị ảnh hưởng bởi class. Trong ví dụ này, chỉ các phần tử <a> với class = "readmore" mới được áp dụng css:

```
a.readmore {
    align-self: flex-end;
    background-color: #1abc9c !important;
    border-radius: 5px;
    color: white;
    padding: 6px 18px;
    text-decoration: none;
}
```

Các phần tử HTML cũng có thể tham chiếu đến nhiều hơn một lớp. Trong ví dụ này, phần tử <**p**> sẽ được tạo kiểu theo cả "readmore" và "dblock":

```
<style>
.readmore {
    align-self: flex-end;
    background-color: #1abc9c !important;
    border-radius: 5px;
    color: white;
    padding: 6px 18px;
    text-decoration: none;
}
.dblock{
    text-align: center;
    display: block;
}
</style>
<p>HTML là ngôn ngữ lập trình dùng để xây dựng và cấu trúc lại các thành phần có trong Website...</p><a class="readmore">Đọc thêm</a>
```

Kết quả hiển thị trên trình duyệt:

Bộ chọn * chọn tất cả các phần tử HTML trên trang. Ví dụ CSS dưới đây sẽ ảnh hưởng đến mọi phần tử HTML trên trang:

```
* {
 text-align: center;
 color: blue;
}
```

Ngoài ra có thể nhóm các Bộ chọn và áp dụng cùng một kiểu CSS. Trong ví dụ này, **h1**, **#title** và **.center** sẽ có cùng 1 kiểu:

```
h1, #title, .center {
 text-align: center;
 color: red;
}
```

3.3.3 Bộ chọn thuộc tính

Bộ chọn [attribute] được sử dụng để chọn các yếu tố với một thuộc tính cụ thể. Ví dụ sau đây chọn tất cả các phần tử **a** có thuộc tính target :

```
a[target] {
 background-color: yellow;
}
```

Bộ chọn [attribute= "value"] được sử dụng để chọn các phần tử với một thuộc tính bằng một giá trị cụ thể nào đó. Ví dụ sau sẽ chọn tất cả các phần tử a với thuộc tính target = "_blank":

```
a[target="_blank"] {
 background-color: yellow;
}
```

Bộ chọn [attribute~= "value"] được sử dụng để chọn các phần tử với một giá trị thuộc tính có chứa một từ nhất định. Ví dụ sau sẽ chọn tất cả các phần tử với một thuộc tính **title** có chứa từ "box":

```
[title~="box"] {
 border: 5px solid yellow;
}
```

Bộ chọn [attribute|= "value"] được sử dụng để chọn các phần tử với các thuộc tính bắt đầu bằng một giá trị nào đó. Ví dụ sau sẽ chọn tất cả các phần tử với một giá trị thuộc tính lớp bắt đầu bằng"yellow". Lưu ý rằng giá trị phải là toàn bộ một từ, hoặc một mình, giống như class="yellow", hoặc theo sau là một dấu gạch ngang (-), giống như class= "yellow-text"

```
[class|="yellow"] {
 background: yellow;
}
```

Bộ chọn [attribute^= "value"] được sử dụng để chọn các phần tử có giá trị thuộc tính bắt đầu bằng một giá trị xác định nào đó. Ví dụ sau sẽ chọn tất cả các phần tử với một giá trị thuộc tính lớp bắt đầu với "top":

```
[class^="top"] {
 background: yellow;
}
```

Bộ chọn [attribute $= "value"] được sử dụng để chọn các phần tử có giá trị thuộc tính kết thúc bằng một giá trị xác định. Ví dụ sau sẽ chọn tất cả các phần tử với một giá trị thuộc tính lớp kết thúc bằng "test":

```
[class$="test"] {
 background: yellow;
}
```

Bộ chọn [attribute *= "value"] được sử dụng để chọn các phần tử có giá trị thuộc tính chứa một giá trị xác định. Ví dụ sau sẽ chọn tất cả các yếu tố với một giá trị thuộc tính lớp có chứa "red" :

```
[class*="red"] {
 background: red;
}
```

Các bộ chọn thuộc tính có thể hữu ích cho các hình thức định dạng mà không cần class hoặc id. Ví dụ chọn tất cả những phần tử biểu mẫu có type là button:

```
input[type=submit] {
    padding: 10px;
    font-size: 18px;
    border: 1px solid #1abc9c;
    background-color: #1abc9c;
    color: white;
    margin-left: 20px;
}
```

Kết quả khi hiển thị nút submit ra màn hình:

3.3.4 Bộ chọn tổ hợp

Bộ chọn tổ hợp (Combinator) thể hiện mối quan hệ giữa các selector, cho phép kết hợp các selector lại với nhau dưới dạng các chuỗi ký tự. Có 4 combinator quen thuộc như sau:

- **Bộ chọn con cháu** (sử dụng dấu khoảng trắng): Chọn những phần tử là con cháu của một phần tử khác.
- **Bộ chọn con (sử dụng dấu >):** Chọn những phần tử là con của một phần tử khác.
- **Bộ chọn anh em liền kề (sử dụng dấu +)**: Select phần tử nằm liền kề và ngang hàng với phần tử khác (các phần tử "anh em" kề nhau).
- **Bộ chọn anh em (sử dụng dấu ~)** : Chọn những phần tử ngang hàng với phần tử khác (các phần tử "anh em").

Chú ý ở đây "con cháu" được dùng để chỉ phần tử nằm trong phần tử khác, còn "con" là chỉ phần tử nằm trực tiếp ngay bên trong của phần tử khác. Ví dụ như ** example ** thì chỉ có span là "con" của <**div**>, còn cả <**span**> và ** ** đều là "con cháu" của <**div**>.

Bộ chọn con cháu (sử dụng dấu khoảng trắng) cho phép kết hợp những phần tử là con cháu của một phần tử khác. Ví dụ chọn tất cả các phần tử <**textarea**> nằm trong phần tử bất kỳ có class là **form-subblock**:

```
.form-subblock textarea {
    border: none;
    height: 100px;
    resize: none;
    font-size: 18px;
    font-family: Arial;
    color: #00896d;
}
```

Bộ chọn con (sử dụng dấu >) cho phép kết hợp những phần tử là con "trực tiếp" của một phần tử khác

```
div > p{
    flex-grow: 1;
    display: flex;
    flex-direction: column;
    padding: 20px;
}
```

Ở đây, phần tử con là thẻ <**p**>. Thẻ <**p**> nằm ở trong thẻ <**div**> sẽ có **background-color**: **yellow**, còn các thẻ con <**p**> khác (nếu có) nằm trong <**p**> thì không được áp dụng.

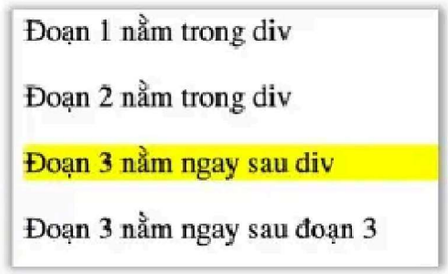
Kết quả hiển thị:

Hình 3-6 Minh họa cách làm việc của bộ chọn anh em liền kề (dấu >)

Bộ chọn anh em liền kề (sử dụng dấu +) cho phép chọn những phần tử nằm liền kề và ngang hàng với phần tử được chỉ định (các phần tử "anh em" kề nhau). Như ví dụ dưới đây, phần tử <**p**> nằm liền kề và ngang hàng với <**div**>sẽ có **background-color: yellow,** còn các thẻ <**p**> khác nằm trong <**div**> hoặc thẻ <**p**> ngang hàng nhưng không liền kề với div thì không:

```
<style>
div + p {
  background-color: yellow;
}
</style>
<div>
  <p>Đoạn 1 nằm trong div</p>
  <p>Đoạn 2 nằm trong div</p>
</div>
<p>Đoạn 3 nằm ngay sau div</p>
<p>Đoạn 3 nằm ngay sau đoạn 3</p>
```

Kết quả hiển thị:



Hình 3-7 Minh họa cách làm việc của bộ chọn anh em liền kề (dấu +)

Bộ chọn anh em (sử dụng dấu ~) cho phép chọn những phần tử nằm ngang hàng với phần tử được chỉ định (các phần tử "anh em").

```
div ~ p {
background-color:yellow;
}
```

Ở đây, phần tử <**p**> nằm ngang hàng với div sẽ có **background-color: yellow**, còn các thẻ <**p**> khác nằm trong <**div**> thì không. Ví dụ:

Hình 3-8 Minh họa cách làm việc của bộ chọn anh em (dấu +)

3.3.5 Bộ chọn lớp giả

Bộ chọn lớp giả được sử dụng để xác định tình trạng đặc biệt của một phần tử. Nó có thể được sử dụng để:

- Định dạng một phần tử khi người dùng lướt chuột qua nó
- Định dạng một phần tử liên kết khi người dùng đã từng vào
- Định dạng một phần tử khi nó được chọn (active)

Cú pháp:

```
selector:pseudo-class {
 property:value;
}
```

Các lớp giả **:hover, :link, :visited, và :active** thường xuyên được sử dụng với liên kết có thể được hiển thị theo những cách khác nhau:

```
a:link {
 color: #FF0000;
}
a:visited {
 color: #00FF00;
}
a:hover {
 color: #FF00FF;
}
a:active {
 color: #0000FF;
}
```

Chú ý: **a:hover** phải sau **a:link** và **a:visited.** Ngoài ra **a:active** phải sau **a:hover** trong định nghĩa CSS.

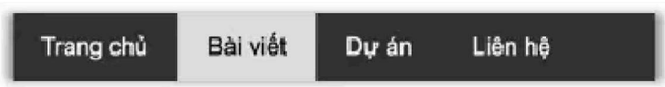
Ví dụ để tạo một thanh điều hiểu hướng (Navigation) có thể viết mã như sau:

```
<nav>
    <a href="index.html">Trang chủ</a>
    <a href="#">Bài viết</a>
    <a href="#">Dự án</a>
    <a href="#">Liên hệ</a>
</nav>
```

Sau đó khai báo các thuộc tính CSS:

```
nav {
    display: flex;
    background-color: #333;
    flex: 100%;
    padding: 0px;
}
nav a {
    color: white;
    padding: 14px 20px;
    text-decoration: none;
    text-align: center;
}
nav a:hover {
    background-color: #ddd;
    color: black;
}
```

Kết quả khi chạy sẽ tạo ra một thanh điều hướng, khi người dùng di chuột vào mỗi tùy chọn thì sẽ có hiệu ứng đổi màu nền:



Hình 3-9 Thanh điều hướng được tạo ra bởi CSS

Lớp giả :first-child so khớp với một phần tử là con đầu của phần tử khác. Ví dụ chọn ra thẻ <**p**> đầu tiên mà không chọn các thẻ <**p**> khác.

```
p:first-child {
 color: blue;
}
```

Tất cả các phần tử <**i**> đầu tiên trong 1 phần tử <**p**>.

```
p i:first-child {
 color: blue;
}
```

Có rất nhiều bộ chọn lớp giả khác. Các bộ chọn lớp giả hay dùng này được mô tả trong bảng dưới đây:

Bảng 3-1 Danh sách các bộ chọn lớp giả khác

Bộ chọn lớp giả	Ví dụ	Mô tả ví dụ
:checked	input:checked	Chọn phần tử <**input**> đã được chọn
:disabled	input:disabled	Chọn phần tử <**input**> bị vô hiệu hóa
:empty	p:empty	Chọn mỗi phần tử <**p**> không có con
:enabled	input:enabled	Chọn mỗi phần tử <**input**> được bật
:invalid	input:invalid	Chọn phần tử <**input**> có giá trị không hợp lệ
:lang(language)	p:lang(it)	Chọn phần tử <**p**> với một giá trị thuộc tính lang bắt đầu với "it"
:last-child	p:last-child	Chọn phần tử <**p**> là con cuối cùng
:not(selector)	:not(p)	Chọn mọi phần tử không phải là một phần tử <**p**>
:nth-child(n)	p:nth-child(2)	Chọn phần tử <**p**> là con thứ ha
:nth-last-child(n)	p:nth-last-child(2)	Chọn phần tử <**p**> là con thứ hai của cha, kể từ con cuối cùng
:only-child	p:only-child	Chọn phần tử <**p**> là con duy nhất
:out-of-range	input:out-of-range	Chọn phần tử <**input**> với một giá trị không nằm trong khoảng
:read-only	input:read-only	Chọn phần tử <**input**> với thuộc tính "readonly"
:read-write	input:read-write	Chọn các phần tử <**input**> mà không có thuộc tính "readonly"

:required	input:required	Chọn phần tử <**input**> với thuộc tính "required"
:root	root	Chọn phần tử gốc của tài liệu.
:valid	input:valid	Chọn tất cả các phần tử <input> với giá trị hợp lệ.

3.3.6 Bộ chọn phần tử giả

Một bộ chọn phần tử giả được sử dụng để xác định ra những phần tử không tồn tại, thường sẽ thuộc một phần tử có thật. Các phần tử giả này hữu dụng để:

- Tạo kiểu cho chữ cái đầu tiên, hoặc dòng của một phần tử
- Chèn nội dung trước, hoặc sau nội dung của một phần tử

Phần tử giả ::first-line được sử dụng để thêm một phong cách riêng cho dòng đầu tiên của văn bản. Ví dụ sau định dạng dòng đầu tiên của văn trong tất cả phần tử <**p**>:

```
p::first-line {
 color: #ff0000;
 font-variant: small-caps;
}
```

Lưu ý: Các phần tử giả **::first-line** chỉ có thể được áp dụng cho các phần tử ở mức khối (block)

Các thuộc tính sau áp dụng cho các phần tử giả **::first-line**:

Các phần tử giả **::first-letter** được sử dụng để thêm một phong cách riêng cho các chữ cái đầu tiên của một văn bản. Ví dụ sau định dạng chữ cái đầu tiên của văn bản trong tất cả các phần tử <**p**>:

```
p::first-letter {
 color: #ff0000;
 font-size: xx-large;
}
```

Lưu ý : Các phần tử giả **::first-letter** chỉ có thể được áp dụng cho các phần tử khối ví dụ như phần tử p. Ví dụ:

```
p.intro::first-letter {
 color: #ff0000;
 font-size:200%;
}
```

Ví dụ trên hiển thị các chữ cái đầu tiên của đoạn văn với **class = "intro"**, màu đỏ và kích thước lớn hơn.

Phần tử giả **::before** có thể được sử dụng để chèn một số nội dung trước nội dung của một phần tử. Ví dụ sau chèn một hình ảnh trước nội dung của mỗi phần tử <**h1**> :

```
h1::before {
 content: url(smiley.gif);
}
```

Phần tử giả **::after** có thể được sử dụng để chèn một số nội dung sau nội dung của một phần tử. Ví dụ sau chèn một hình ảnh sau nội dung của phần tử <**h1**> :

```
h1::after {
 content: url(smiley.gif);
}
```

Phần tử giả **::selection** xác định các nội dung được lựa chọn bởi người dùng. Các thuộc tính CSS sau đây có thể được áp dụng cho phần tử giả ::selection:

- color
- background
- cursor
- outline

Ví dụ sau đây làm cho các văn đã chọn màu đỏ trên nền màu vàng:

```
::selection {
 color: red;
 background: yellow;
}
```

3.4 SỬ DỤNG MÀU SẮC

Màu trong CSS có thể được xác định bằng cách sử dụng:

- Mã màu **Hexadecimal**: Sử dụng mã màu hệ 16
- Hàm **rgb**(): Sử dụng hệ màu RGB
- Hàm **hsl**(): Sử dụng hệ màu HSL
- Tên **màu**: Sử dụng tên để xác định màu (có 140 tên)

3.4.1 Màu hệ thập lục

Màu **hệ thập lục (Hexadecimal)** sử dụng mã màu gồm có các thành phần RGB. Ví dụ:

```
#p1 {background-color: #ff0000;}  /* red */
#p2 {background-color: #00ff00;}  /* green */
#p3 {background-color: #0000ff;}  /* blue */
```

Một màu **Hexadecimal** được quy định với cú pháp #RRGGBB, trong đó các số nguyên **Hexadecimal** RR(màu đỏ), GG(màu xanh) và BB(màu xanh dương) chỉ định các thành phần của màu. Tất cả các giá trị phải nằm trong khoảng từ 00 đến FF (theo hệ 16). Ví dụ, giá trị #0000ff được hiển thị là màu xanh lam, vì thành phần màu xanh lam được đặt thành giá trị cao nhất(ff) và các giá trị khác được đặt thành 00.

Hình 3-10 Bảng một số màu sử dụng mã màu thập lục

3.4.2 RGB

Màu RGB có thể xác định bằng hàm **rgb**(), giá trị cho mỗi màu từ 0 đến 255. Ví dụ:

```
#p1 {background-color: rgb(255, 0, 0);}  /* Màu đỏ */
#p2 {background-color: rgb(0, 255, 0);}  /* Màu xanh lá */
#p3 {background-color: rgb(0, 0, 255);}  /* Màu xanh lam */
```

Giá trị màu **RGB** được chỉ định với hàm **rgb**(), có cú pháp sau: rgb(đỏ, xanh lá cây, xanh dương). Mỗi tham số (đỏ, lục và lam) xác định cường độ của màu là một số nguyên trong khoảng từ 0 đến 255 hoặc giá trị phần trăm (từ 0% đến 100%). Ví dụ: giá trị rgb(0,0,255) được hiển thị là màu xanh lam, bởi vì tham số màu xanh được đặt thành giá trị cao nhất của nó (255) và các giá trị khác được đặt thành 0. Các giá trị sau xác định màu bằng nhau: rgb(0,0,255) và rgb (0%, 0%, 100%). RGB có thể sử dụng thêm **alpha** - chỉ định độ trong suốt của đối tượng. Tham số **alpha** là một số trong khoảng 0,0 (hoàn toàn trong suốt) và 1,0 (hoàn toàn mờ). Ví dụ:

```
#p1 {background-color: rgba(255, 0, 0, 0.3);} /* Màu đỏ, trong suốt 0.3 */
#p2 {background-color: rgba(0, 255, 0, 0.3);} /* Màu xanh lá, trong suốt 0.3 */
#p3 {background-color: rgba(0, 0, 255, 0.3);} /* Màu xanh lam, trong suốt 0.3 */
```

3.4.3 HSL

HSL là một hệ màu phổ biến được sử dụng để xác định màu sắc, để sử dụng hệ màu HSL cần sử dụng hàm hsl()

```
<h2 style="background-color:hsl(0, 100%, 50%);">hsl(0, 100%, 50%)</h2>
<h2 style="background-color:hsl(240, 100%, 50%);">hsl(240, 100%, 50%)</h2>
<h2 style="background-color:hsl(147, 50%, 47%);">hsl(147, 50%, 47%)</h2>
<h2 style="background-color:hsl(300, 76%, 72%);">hsl(300, 76%, 72%)</h2>
<h2 style="background-color:hsl(39, 100%, 50%);">hsl(39, 100%, 50%)</h2>
<h2 style="background-color:hsl(248, 53%, 58%);">hsl(248, 53%, 58%)</h2>
```

HSL là viết tắt của màu sắc, độ bão hòa và độ sáng. Giá trị màu HSL được chỉ định với hàm hsl(), có cú pháp sau: hsl(hue, saturation, light). Hue là độ trên bánh xe màu (từ 0 đến 360) - 0 (hoặc 360) là màu đỏ, 120 là màu xanh lá cây, 240 là màu xanh. Saturation là một giá trị phần trăm; 0% có nghĩa là một màu xám và 100% là màu đầy đủ. Light cũng là tỷ lệ phần trăm; 0% là màu đen, 100% là màu trắng. Các giá trị màu HSLA là phần mở rộng của các giá trị màu HSL với kênh alpha - chỉ định độ mờ của đối tượng.

3.4.4 Tên màu

Trong CSS, một màu có thể được chỉ định bằng cách sử dụng tên màu.

Hình 3-11 Màu sắc và tên màu

Ví dụ dùng tên màu cho màu nền:

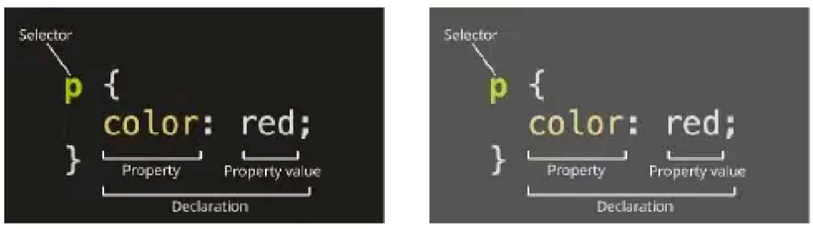
```
<h1 style="background-color:Tomato;">Hello World</h1>
<p style="background-color:Orange;">Lorem ipsum...</p>
```

Dùng tên màu cho màu chữ:

```
<h1 style="color: DodgerBlue;">Hello World</h1>
<p style="color:Gray;">Lorem ipsum...</p>
<p style="color:Violet;">Ut wisi enim...</p>
```

3.4.5 Độ trong suốt

Các thuộc tính **opacity** quy định các tính trong suốt của một phần tử. Các thuộc tính **opacity** có thể mất một giá trị 0.0-1.0. Giá trị càng thấp, tính trong suốt càng cao:



Hình 3-12 Hình ảnh với opacity tương ứng là 1 và 0.75

Thuộc tính **opacity** thường được sử dụng với các bộ chọn **:hover** để thay đổi độ mờ đục khi di chuột qua:

```
article content img:hover {
    opacity: .75;
}
```

Khi sử dụng thuộc tính **opacity** để thêm tính trong suốt cho nền tảng của một phần tử, tất cả các phần tử con của nó kế thừa tính trong suốt giống vậy. Điều này có thể làm cho văn bên trong một phần tử hoàn toàn trong suốt khó đọc. Nếu không muốn áp dụng **opacity** cho phần

tử con, có thể sử dụng giá trị màu RGBA. Ví dụ sau áp dụng **opacity** cho màu nền và không áp dùng cho phần văn bản:

```
div {
 background: rgba(76, 175, 80, 0.3)
}
```

3.5 ĐỊNH DẠNG NỀN

3.5.1 Màu nền

Thuộc tính **background-color** quy định màu nền của một phần tử. Ví dụ đoạn mã sau sẽ làm thay đổi màu nền của <**header**>:

```
</style>
header {
    background: #1abc9c;
    color: white;
    padding: 60px;
    text-align: center;
}
</style>
<header>
    <h1>Minh's Page</h1>
    <p>Lập kế hoạch - Ghi chú thích - <b>Viết mã lệnh</b> - Kiểm thử</p>
</header>
```

Kết quả hiển thị:

3.5.2 Hình nền

Thuộc tính **background-image** quy định cụ thể một hình ảnh nền cho các phần tử HTML. Theo mặc định, hình ảnh được lặp lại để nó bao phủ toàn bộ phần tử. Ví dụ:

```
header{
    color: white;
    padding: 60px;
    text-align: center;
    background-image: url('/public/images/1058.png');
}
```

Theo mặc định, thuộc tính **background-image** lặp lại một hình ảnh theo cả chiều ngang chiều dọc.

Để lặp lại theo 1 chiều cụ thể nào đó, sử dụng giá trị **repeat-x** hoặc **repeat-y**. Để cho hình nền không lặp lại, sử dụng **no-repeat**. Đối với các hình ảnh không lặp lại, mà muốn vừa khít với hộp chứa nó, nên sử dụng thuộc tính **background-size:cover** như sau:

```
header {
    color: white;
    padding: 60px;
    text-align: center;
    background-image: url('/public/images/feature.jpg');
    background-size:auto ;
}
```

3.5.3 Vị trí hình nền

Thuộc tính **background-position** được sử dụng để xác định vị trí của các hình nền. Ví dụ, nếu định vị hình ảnh nền ở góc trên bên phải:

```
header {
    color: white;
    padding: 60px;
    text-align: center;
    background-image: url('/public/images/feature.jpg');
    background-position: right top;
}
```

3.5.4 Hình nền cố định

Thuộc tính **background-attachment** cho phép hình nền nên di chuyển hoặc được cố định (sẽ không di chuyển với phần còn lại của trang).

```
header {
    color: white;
    padding: 60px;
    text-align: center;
    background-color: #1abc9c;
    background-image: url('/public/images/feature.jpg');
    background-repeat: no-repeat;
    background-attachment: fixed;
    background-position: top left;
}
```

3.5.5 Cách viết rút gọn

Để rút ngắn mã, cũng có thể chỉ định tất cả các thuộc tính nền trong một thuộc tính **background** duy nhất. Khi sử dụng cách rút gọn này, thứ tự của các giá trị thuộc tính là: background-color, background-image, background-repeat, background-attachment, background-position. Ví dụ:

```
header {
    color: white;
    padding: 60px;
    text-align: center;
    background: #1abc9c url('/public/images/feature.jpg') no-repeat fixed top
left;
}
```

3.6 ĐỊNH DẠNG VIỀN

3.6.1 Kiểu viền

Các **border-style** quy định cụ thể những loại border để hiển thị. Các giá trị thông dụng khi biểu diễn đường viền:

- dotted - Xác định đường viền chấm
- dashed - Xác định đường viền nét đứt
- solid - Xác định đường viền liền
- double - Xác định đường viền đôi
- none - Xác định không có border
- hidden - Xác định đường viền ẩn

Định dạng viền có thể có từ một đến bốn giá trị (trên, phải, dưới, và trái). Ví dụ: Các kiểu viền khác nhau:

```
p.dotted {border-style: dotted;}
p.dashed {border-style: dashed;}
p.solid {border-style: solid;}
p.double {border-style: double;}
```

```
p.groove {border-style: groove;}
p.ridge {border-style: ridge;}
p.inset {border-style: inset;}
p.outset {border-style: outset;}
p.none {border-style: none;}
p.hidden {border-style: hidden;}
p.mix {border-style: dotted dashed solid double;}
```

3.6.2 Độ rộng đường viền CSS

Thuộc tính **border-width** chỉ định chiều rộng của border. Ví dụ:

```
p.one {
 border-style: solid;
 border-width: 5px;
}
p.two {
 border-style: solid;
 border-width: medium;
}
p.three {
 border-style: solid;
 border-width: 2px 10px 4px 20px;
}
```

Chiều rộng có thể được đặt thành một kích thước cụ thể (tính bằng px, pt, cm, em, v.v...) hoặc bằng cách sử dụng một trong ba giá trị được xác định trước: mỏng, trung bình hoặc dày. Các border-width có thể có từ một đến bốn giá trị (trên, phải, dưới, và trái).

3.6.3 Màu CSS

Thuộc tính **border-color** được sử dụng để thiết lập màu sắc của viền. Ví dụ:

```
p.one {
 border-style: solid;
 border-color: red;
}
p.two {
 border-style: solid;
 border-color: green;
}
p.three {
 border-style: solid;
 border-color: red green blue yellow;
}
```

3.6.4 Border từng phía

Trong CSS, cũng có các thuộc tính định kiểu viền cho từng phía (trên, phải, dưới và trái):

- **border-top-style:** Viền trên
- **border-right-style:** Viền phải
- **border-bottom-style:** Viền dưới

- **border-left-style:** Viền trái

```
p {
 border-top-style: dotted;
 border-right-style: solid;
 border-bottom-style: dotted;
 border-left-style: solid;
}
```

3.6.5 Cách viết rút gọn

Để rút ngắn mã, cũng có thể chỉ định tất cả các thuộc tính viền riêng lẻ trong một thuộc tính **border** . Ví dụ thiết lập cho các thẻ p có viền 5px:

```
p {
 border: 5px solid red;
}
```

3.6.6 Viền bo tròn

Thuộc tính **border-radius** được sử dụng để tạo viền với bo cong ở góc:

```
p {
 border: 2px solid red;
 border-radius: 5px;
}
```

3.7 ĐỊNH DẠNG VĂN BẢN

3.7.1 Màu sắc

Sử dụng thuộc tính **color** để xác định màu cho văn bản. Ví dụ:

```
body {
 color: blue;
}
h1 {
 color: green;
}
```

3.7.2 Căn chỉnh văn bản

Thuộc tính **text-align** để căn (trái, phải, giữa) cho văn bản:

```
h1 {
 text-align: center;
}
h2 {
 text-align: left;
}
h3 {
 text-align: right;
}
```

Ngoài ra, khi text-align được đặt thành "justify", mỗi dòng được kéo dài sao cho mỗi dòng có chiều rộng bằng nhau

```
div {
  text-align: justify;
}
```

3.7.3 Trang trí văn bản

Thuộc tính **text-decoration** được sử dụng để thiết lập (hoặc loại bỏ) trang trí văn bản. Giá trị text-decoration: none; thường được sử dụng để xóa gạch chân khỏi các liên kết. Các giá trị khác được sử dụng để trang trí văn bản. Ví dụ:

```
h2 {
  text-decoration: overline;
}

h3 {
  text-decoration: line-through;
}

h4 {
  text-decoration: underline;
}
```

Lưu ý: Không nên gạch chân văn không phải là một liên kết, vì điều này thường gây nhầm lẫn cho người đọc

3.7.4 Chuyển đổi văn bản

Thuộc tính **text-transform** được sử dụng để xác định chữ hoa và chữ thường trong một văn bản. Ví dụ:

```
p.uppercase {
 text-transform: uppercase;
}
p.lowercase {
 text-transform: lowercase;
}
p.capitalize {
 text-transform: capitalize;
}
```

3.7.5 Thụt đầu dòng

Thuộc tính **text-indent** được dùng để xác định thụt đầu dòng của dòng đầu tiên của một văn bản. Ví dụ

```
p {
 text-indent: 50px;
}
```

3.7.6 Khoảng cách chữ

Các **letter-spacing** được sử dụng để xác định khoảng cách giữa các ký tự trong một văn bản. Ví dụ sau minh họa cách tăng hoặc giảm khoảng trắng giữa các ký tự:

```
h1 {
 letter-spacing: 3px;
}
h2 {
 letter-spacing: -3px;
}
```

3.7.7 Khoảng cách từ

Các **word-spacing** được sử dụng để xác định khoảng cách giữa các từ trong một văn. Ví dụ sau minh họa cách tăng hoặc giảm khoảng trắng giữa các từ:

```
h1 {
 word-spacing: 10px;
}
h2 {
 word-spacing: -5px;
}
```

3.7.8 Chiều cao giữa các dòng

Thuộc tính **line-height** được dùng để tăng/giảm khoảng cách giữa các dòng. Giá trị mặc định của line-height là 1.2, có thể thay đổi như sau:

```
p.small {
 line-height: 0.8;
}
p.big {
 line-height: 1.8;
}
```

3.7.9 Hướng văn bản

Thuộc tính **direction** được sử dụng để thay đổi hướng văn bản. Ví dụ, để văn bản được viết theo chiều từ phải sang trái:

```
p {
 direction: rtl;
}
```

3.7.10 Bóng văn bản

Thuộc **text-shadow** thêm bóng cho văn bản. Ví dụ sau chỉ định vị trí của bóng ngang (3px), vị trí của bóng dọc (2px) và màu bóng là màu đỏ:

```
h1 {
 text-shadow: 3px 2px red;
}
```

3.7.11 Font chữ

Để thiết lập Font cho một trang HTML hoặc cho từng phần tử trong trang, sử dụng các thuộc tính sau:

- **font-family**: được sử dụng để thay đổi bề mặt font
- **font-style**: được sử dụng để tạo một font chữ nghiêng hoặc chếch
- **font-variant**: được sử dụng để tạo những chữ hoa nhỏ (small-cap)
- **font-weight**: được sử dụng để tăng giảm độ đậm của font
- **font-size**: được sử dụng để xác định kích cỡ font
- **font**: rút gọn việc sử dụng nhiều thuộc tính

Thuộc tính **font-family** được sử dụng để lựa chọn font chữ. Các font chữ được hỗ trợ bởi hầu hết các trình duyệt.

- Arial (sans-serif)
- Verdana (sans-serif)
- Helvetica (sans-serif)
- Tahoma (sans-serif)
- Trebuchet MS (sans-serif)
- Times New Roman (serif)
- Georgia (serif)
- Garamond (serif)
- Courier New (monospace)
- Brush Script MT (cursive)

Khi sử dụng thuộc tính **font-family**, nên xác định nhiều hơn một giá trị. Giá trị cho thuộc tính font-family là bất kỳ tên **font-family** hợp lệ nào. Nếu tên font có dấu cách, thì phải đặt trong trích dẫn kép, ví dụ như: "Times New Roman". Nếu trình duyệt không hỗ trợ giá trị font đầu tiên, nó sẽ thử giá trị tiếp theo, … Bắt đầu với giá trị font mà muốn và kết thúc với một generic family để trình duyệt chọn một font tương tự trong generic family nếu không có font nào có sẵn. Thuộc tính **font-family** phải giữ một số tên phông chữ dưới dạng hệ thống "dự phòng", để đảm bảo khả năng tương thích tối đa giữa các trình duyệt / hệ điều hành. Nếu trình duyệt không hỗ trợ phông chữ đầu tiên, nó sẽ thử phông chữ tiếp theo. Ví dụ:

```
p {
 font-family: "Times New Roman", Times, serif;
}
```

Thuộc tính **font-style** trong CSS thường được sử dụng khi muốn xác định một font chữ nghiêng. Các giá trị mà thuộc tính này có thể nhận là: **normal**, **italic** hoặc **oblique**. Trong đó, **normal** là

hiển thị văn như bình thường, **italic** và **oblique** thì khá giống nhau, để hiển thị dạng nghiêng. Tuy nhiên, **oblique** ít được hỗ trợ hơn. Ví dụ

```
p.normal {
 font-style: normal;
}
p.italic {
 font-style: italic;
}
p.oblique {
 font-style: oblique;
}
```

Thuộc tính **font-variant** nhận hai giá trị là **normal** và **small-caps**, để tạo ra chữ thường hoặc chữ hoa nhỏ

```
p.normal {
 font-variant: normal;
}
p.small {
 font-variant: small-caps;
}
```

Thuộc tính **font-weight** tạo ra chữ đậm, nhạt. Thuộc tính này có thể nhận các giá trị: normal, bold, bolder, lighter, 100, 200, 300, 400, 500, 600, 700, 800, 900.

```
p.normal {
 font-weight: normal;
}p.thick {
 font-weight: bold;
}
```

Để thiết lập kích cỡ chữ, sử dụng thuộc tính **font-size**. Thuộc tính này có thể nhận các giá trị tương đối hoặc tuyệt đối: **xx-small**, **x-small**, **small**, **medium**, **large**, **x-large**, **xx-large**, **smaller**, **larger**, **px, in,** và **%**.Ví dụ:

```
<p style="font-size:20px;">Doan van nay co Font Size la 20 pixel.</p>
<p style="font-size:small;">Doan van nay co Font Size la small</p>
<p style="font-size:large;">Doan van nay co Font Size la large</p>
```

Để cho phép người dùng tăng giảm kích cỡ của văn bản (trong menu của trình duyệt), nhiều lập trình viên sử dụng đơn vị **em** thay cho **pixel**. Đơn vị **em** được đề nghị sử dụng bởi W3C. **1em** thì bằng với kích cỡ font hiện tại. Kích cỡ văn mặc định là **16px**, do đó kích cỡ mặc định của 1em sẽ là **16px**.

Để tối thiểu hóa lượng code cần phải viết, có thể sử dụng thuộc tính font trong CSS để thiết lập tất cả thuộc tính liên quan tới Font đã được trình bày ở phần trên. Trong thuộc tính **font** này, thiết lập các thuộc tính liên quan tới font theo thứ tự như trong cú pháp sau:

```
font-style font-variant font-weight font-size/line-height font-family
```

Trong đó, giá trị của hai thuộc tính **font-size** và **font-family** là bắt buộc. Nếu không xác định một trong các giá trị khác thì giá trị mặc định sẽ được sử dụng (nếu có). Ví dụ đơn giản về cách sử dụng thuộc tính font trong CSS:

```
<p style="font:italic small-caps bold 15px georgia;">Văn bản</p>
```

3.8 ĐỊNH DẠNG LIÊN KẾT

Để định dạng các liên kết, chẳng hạn như tạo màu, font, … cho các đường link trong CSS, có thể sử dụng nhiều thuộc tính CSS đa dạng, ví dụ như **color**, **font-family**, **background**, … Ví dụ:

```
a {
color: hotpink;
}
```

Ngoài ra, để thiết lập các trạng thái khác nhau cho link, có thể sử dụng các thuộc tính:

- **:link** – biểu thị rằng trang Web này trình duyệt chưa lưu (tức người sử dụng lần đầu tiên click vào đường dẫn này).
- :**visited** – biểu thị rằng đường dẫn tới trang Web này đã được lưu bởi trình duyệt (tức là người sử dụng đã click vào đường dẫn này trước đó rồi).
- **:hover** – biểu thị rằng khi người sử dụng di chuyển chuột qua phần tử mà chứa link đó (tức là phần tử đó là một link khi người sử dụng di chuyển chuột qua phần tử đó).
- **:active** – biểu thị đường link là active khi người sử dụng click chuột vào.

Ví dụ:

```
a:link {color: #000000}
a:visited {color: #006600}
a:hover {color: #FFCC00}
a:active {color: #FF00CC}
```

Chú ý trạng thái :hover PHẢI theo sau các trạng thái :link và :visited trong định nghĩa CSS thì mới có hoạt động. Trạng thái :active PHẢI theo sau trạng thái :hover trong định nghĩa CSS.

Nếu để ý các ví dụ trên, sẽ thấy rằng các đường link trên đều có dấu gạch chân bên dưới. Vậy làm cách nào để làm biến mất dấu gạch chân này cho đường link, sử dụng thuộc tính **text-decoration** với giá trị là none. Ví dụ:

```
 <style>
a:link {
 text-decoration: none;
}
a:visited {
 text-decoration: none;
}
a:hover {
 text-decoration: underline;
}
a:active {
 text-decoration: underline;
}
</style>
<p><b><a href="default.asp" target="_blank">This is a link</a></b></p>
```

```
<p><b>Note:</b> a:hover MUST come after a:link and a:visited in the CSS definiti
on in order to be effective.</p>
<p><b>Note:</b> a:active MUST come after a:hover in the CSS definition in order
to be effective.</p>
```

Ngoài ra, có thể kết hợp nhiều thuộc tính khác nhau trong CSS để làm cho Link của giống như một Box, hoặc một Button. Ví dụ:

```
a:link, a:visited {
 background-color: #f44336;
 color: white;
 padding: 14px 25px;
 text-align: center;
 text-decoration: none;
 display: inline-block;
}
a:hover, a:active {
 background-color: red;
}
```

3.9 DANH SÁCH

Một danh sách có kèm theo các dấu đầu dòng hoặc dưới đánh số giúp phần văn của dễ đọc và tạo cảm giác thân thiện hơn. Để tạo định dạng riêng cho phần danh sách này, có thể sử dụng các thuộc tính trong CSS. Có 5 thuộc tính trong CSS:

- Thuộc tính **list-style-type** cho phép kiểm soát hình dạng hay bề ngoài của các dấu đầu dòng (giống như Bullet) chẳng hạn ở dạng hình tròn, hình vuông, hay dạng số.
- Thuộc tính **list-style-position** xác định rằng các dấu đầu dòng nên xuất hiện bên trong hay bên ngoài luồng hiển thị nội dung.
- Thuộc tính **list-style-image** sử dụng một hình ảnh để làm các dấu đầu dòng
- Thuộc tính **marker-offset** xác định khoảng cách giữa một dấu đầu dòng và phần văn bản trong danh sách.
- Thuộc tính **list-style** sử dụng thuộc tính này, có thể khai báo một lần mà vẫn có thể xác định được tất cả các thuộc tính trên.

Để điều khiển hình dạng (hình tròn, hình vuông, …) của dấu đầu dòng trong trường hợp một danh sách không có thứ tự và định dạng của các ký tự số (số La Mã, số tự nhiên, …) trong trường hợp danh sách đã được sắp xếp, sử dụng thuộc tính **list-style-type .** Dưới đây là bảng các giá trị có thể được sử dụng cho dạng danh sách không có thứ tự:

Giá trị	Miêu tả
none	Không hiển thị
disc (default)	Một dấu chấm tròn
circle	Một vòng tròn trống
square	Một hình vuông

Dưới đây là bảng các giá trị có thể được sử dụng cho dạng danh sách có sắp xếp:

Giá trị	Miêu tả	Ví dụ
decimal	Số tự nhiên	1,2,3,4,5
decimal-leading-zero	Dạng số, bắt đầu từ 01	01, 02, 03, 04, 05
lower-alpha	Dạng chữ cái thường	a, b, c, d, e
upper-alpha	Dạng chữ cái hoa	A, B, C, D, E
lower-roman	Dạng số La Mã thường	i, ii, iii, iv, v
upper-roman	Dạng số La Mã hoa	I, II, III, IV, V
lower-greek	Dạng chữ Hy Lạp thường	alpha, beta, gamma

Để xác định xem dấu nên hiển thị ở bên trong hay bên ngoài luồng hiển thị nội dung, sử dụng thuộc tính **list-style-position** . Thuộc tính này có thể nhận giá trị: inside hoặc outside. Thuộc tính này có giá trị outside là mặc định khi đó các dấu sẽ nằm bên ngoài danh sách. Ví dụ:

```
ul.a {
  list-style-position: outside;
}

ul.b {
  list-style-position: inside;
}
```

Nếu muốn tạo điểm nhấn cho phần danh sách, có thể sử dụng một hình ảnh để sử dụng thay cho các dấu đơn điệu có sẵn. Để thực hiện điều này, sử dụng thuộc tính **list-style-image** . Cú pháp của thuộc tính này tương tự như thuộc tính **background-image**, tại đây, cần xác định URL tới nơi lưu giữ hình ảnh. Lưu ý rằng, nếu không tìm thấy hình ảnh, thì các bullet mặc định sẽ được sử dụng.

Nếu cảm thấy khoảng cách mặc định giữa marker và phần nội dung hiển thị là dài hơn hoặc ngắn hơn mong muốn có thể sử dụng thuộc tính **marker-offset**.

Sử dụng thuộc tính **list-style** có thể xác định tất cả style cho danh sách (toàn bộ các thuộc tính ở trên) chỉ trong một khai báo duy nhất. Thứ tự của giá trị các thuộc tính này có thể là tùy ý.

Về việc đánh số cho danh sách, có thể sử dụng CSS counters. Đây là biến duy trì bởi CSS có giá trị có thể được tăng lên bởi các quy tắc CSS (để theo dõi bao nhiêu lần chúng được sử dụng). Bộ đếm này cho phép điều chỉnh sự xuất hiện của nội dung dựa trên vị trí của nó trong tài liệu. CSS Counters cũng giống như "biến"

Để làm việc với CSS counters, chúng ta sẽ sử dụng các thuộc tính sau:

Chương 1 **counter-reset** – tạo ra hoặc reset bộ đếm

Chương 2 **counter-increment** – tăng giá trị truy cập

Chương 3 **content** – chèn một nội dung.

Để sử dụng CSS counter, trước tiên nó phải được tạo với counter-reset. Ví dụ dưới đây tạo một bộ đếm cho một trang, sau đó tăng giá trị truy cập cho mỗi phần tử <**h2**> và thêm "Section

<value of the counter>:" vào đầu mỗi phần tử <**h2**> . Phương thức counter() để thêm giá trị của bộ đếm vào một phần tử:

```
body {
 counter-reset: section;
}
h2::before {
 counter-increment: section;
 content: "Section " counter(section) ": ";
}
```

Ví dụ sau đây tạo một bộ đếm cho trang (section) và một bộ đếm cho mỗi phần tử < **h1**> (subsection) . Bộ đếm "section" sẽ được đếm cho từng phần tử <**h1**> với "Section <value of the section counter>" và bộ đếm "subsection" sẽ được đếm cho mỗi phần tử <**h2**> với "<value of the section counter>.<value of the subsection counter>" .

```
body {
 counter-reset: section;
}
h1 {
 counter-reset: subsection;
}

h1::before {
 counter-increment: section;
 content: "Section " counter(section) ". ";
}
h2::before {
 counter-increment: subsection;
 content: counter(section) "." counter(subsection) " ";
}
```

Một bộ đếm cũng có thể hữu ích để tạo danh sách được phác thảo vì một phiên mới của bộ đếm được tạo ra tự động trong các phần tử con. Ở đây chúng ta sử dụng hàm counters() để chèn một chuỗi giữa các mức khác nhau của các bộ đếm lồng nhau:

```
ol {
 counter-reset: section;
 list-style-type: none;
}
li::before {
 counter-increment: section;
 content: counters(section,".") " ";
}
```

3.10 BẢNG

Bảng là một công cụ hiển thị dữ liệu rõ ràng và hiệu quả. Mặc dù với các cách làm mới, thì việc hiển thị dữ liệu bằng các thẻ div thường được sử dụng hơn. Tuy nhiên, với các Website nhỏ và với lượng dữ liệu hiển thị là không lớn thì sử dụng bảng là khá tiện lợi. Phần này sẽ trình bày cách sử dụng các thuộc tính khác nhau trong CSS để làm cho bảng của đẹp hơn. Dưới đây là một số thuộc tính trong CSS:

- Thuộc tính **border** được sử dụng để thiết lập đường viền cho bảng.
- Thuộc tính **border-collapse** xác định rằng các đường viền của bảng nên được vào hợp thành một đường viền.
- Thuộc tính **caption-side** được sử dụng trong phần tử <caption>. Theo mặc định, chúng sẽ được hiển thị ở phần bên trên của bảng. Sử dụng thuộc tính này, bạn có thể xác định vị trí hiển thị của phần tử caption này.
- Thuộc tính **empty-cells** xác định xem có hiển thị đường viền không nếu một ô là trống.
- Thuộc tính **table-layout** cho phép bạn thiết lập layout cho bảng.
- Thuộc tính **width** và **height** để xác định độ rộng và chiều cao. Hai thuộc tính này có thể nhận các giá trị là % hoặc px.

Sử dụng thuộc tính **text-align** để xác định sự căn chỉnh nội dung theo chiều ngang của bảng. Thuộc tính này có thể nhận các giá trị left, right hoặc center. Ví dụ:

```
th {
 text-align: left;
 }
```

Để căn chỉnh bảng theo chiều dọc, sử dụng thuộc tính **vertical-align.** Thuộc tính này có thể nhận các giá trị top, bottom, middle. Theo mặc định thì nội dung của bảng có căn chỉnh dọc với giá trị middle. Ví dụ:

```
td {
 height: 50px;
 vertical-align: bottom;
 }
```

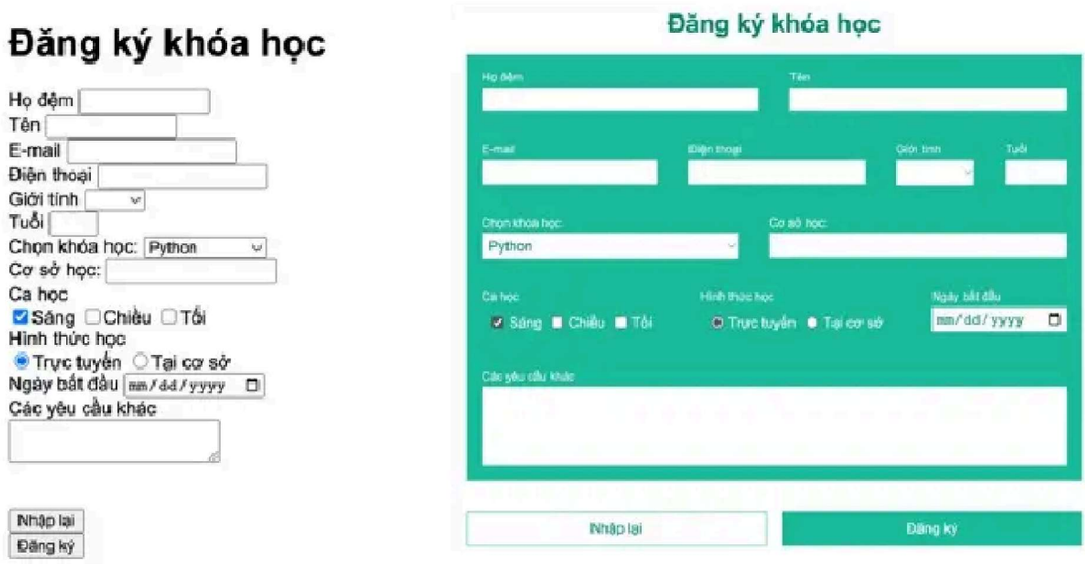
Để điều khiển khoảng cách giữa đường viền và nội dung của bảng, bạn sử dụng thuộc tính **padding**. Giá trị có thể nhận của thuộc tính này là ở dưới dạng đơn vị px. Ví dụ

```
th, td {
 padding: 15px;
 text-align: left;
 }
```

Thuộc tính **border-spacing** xác định khoảng cách giữa các đường viền của các ô trong bảng. Thuộc tính này có thể nhận một hoặc hai giá trị (có đơn vị là độ dài). Nếu cung cấp một giá trị, thì giá trị này sẽ được áp dụng cho đường viền ngang và dọc. Nếu cung cấp hai giá trị, thì tương ứng theo thứ tự sẽ áp dụng cho đường viền ngang và đường viền dọc.

3.11 BIỂU MẪU

Diện mạo của một form HTML có thể được cải thiện rất nhiều với CSS. Ví dụ



Hình 3-13 Biểu mẫu chưa được định dạng và đã được định dạng bởi CSS

Sử dụng thuộc tính **width** để xác định độ rộng của trường input.

```
input {
width: 100%;
}
```

Ví dụ trên áp dụng cho tất cả các phần tử **input**. Nếu chỉ muốn định dạng một kiểu input xác định, có thể sử dụng bộ chọn các thuộc tính:

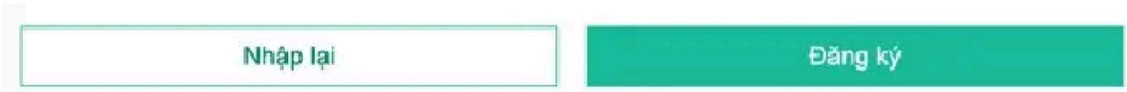
- input[type=text] - chỉ chọn trường văn .
- input[type=password] - chỉ chọn trường mật khẩu.
- input[type=number] - chỉ chọn trường số.

Sử dụng thuộc tính **padding** để thêm không gian bên trong trường văn. Khi có nhiều đầu vào phía sau, cũng có thể muốn thêm thuộc tính **margin**, để thêm không gian bên ngoài chúng. Các thuộc tính khác như background, color, border, giúp trang trí cho các phần tử biểu mẫu được đẹp hơn. Ví dụ đối 2 nút **submit** và **reset** sau:

```
input[type=submit] {
    padding: 10px;
    font-size: 18px;
    border: 1px solid #1abc9c;
    background-color: #1abc9c;
    color: white;
    margin-left: 20px;
}

input[type=reset] {
    padding: 10px;
    font-size: 18px;
    border: 1px solid #1abc9c;
```

```
    background-color: white;
    color: #00896d;
}
```

CSS trên khi áp dụng cho 2 nút submit và reset và form, kết quả như sau:

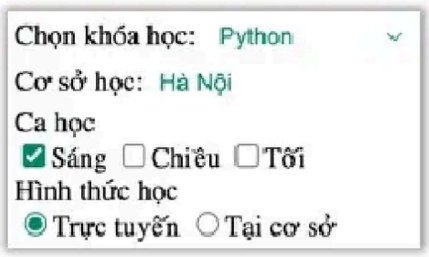


Hình 3-14 Định dangk các nút bấm sau khi áp dụng CSS

Sử dụng thuộc tính **accent-color** để thiết lập màu cở bản cho các điều khiển, ví dụ:

```
.form-block input, select {
    padding: 5px;
    border: 0px;
    color: #00896d;
    accent-color: #00896d;
}
```

Kết quả khi áp dụng cho một số điều khiển sẽ có kết quả như sau:

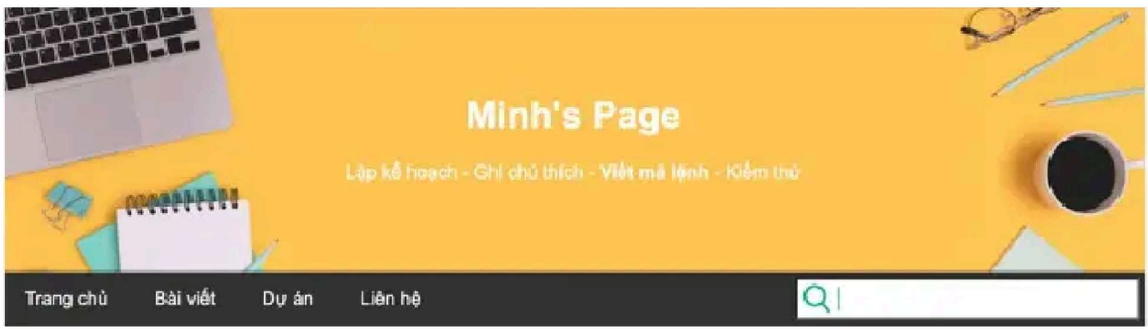


Sử dụng thuộc tính thuộc tính **color** để thay đổi màu sắc văn bản. Ví dụ với màu chủ đạo của trang Web là màu xanh, có thể đổi màu chữ của các ô nhập thành màu xanh cho phù hợp:

Theo mặc định, một số trình duyệt sẽ thêm một đường viền màu xanh xung quanh phần tử input khi nó được focus (nhấp vào). Có thể loại bỏ hành vi này bằng cách thêm **outline:none**; vào input. Sử dụng các :focus selector để thay đổi hiển thị của các input khi nó được người sử dụng lựa chọn (bằng chuột hoặc bàn phím):

```
.form-block input, select:focus {
    outline-width: 0;
}
```

Nếu muốn có một biểu tượng bên trong input, sử dụng thuộc tính **background-image** và đặt nó với thuộc tính **background-position**. Lưu ý rằng cần thêm một phần **padding** lớn bên trái để dành chỗ cho biểu tượng. Ví dụ với ô tìm kiếm có biểu tượng search như dưới đây:



Có thể viết mã như trên để tạo ra ô tìm kiếm:

```
nav div.search input {
    outline: none;
    font-size: 18px;
    background-image: url('/public/images/search.png');
    background-repeat: no-repeat;
    background-size: 24px 24px;
    background-position: 5px center;
    padding-left: 35px;
    color: #00a886;
}
```

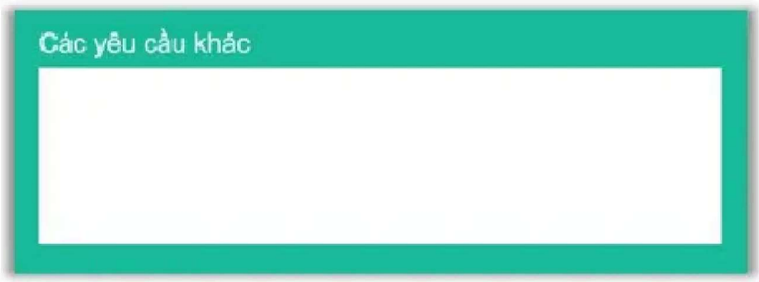
Cũng có thể tạo ra những phần tử nhập có hiệu ứng chuyển động. Trong ví dụ này chúng ta sử dụng thuộc tính **transition** để thay đổi chiều rộng của ô tìm kiếm khi người dùng chọn ô nhập này để tiến hành tìm kiếm.

```
nav div.search input:focus {
    width: 100%;
}
```

Đối với các vùng nhập (textarea). Sử dụng các thuộc tính **resize** để ngăn chặn vùng nhập này khỏi bị thay đổi kích thước bởi người sử dụng:

```
.form-subblock textarea {
    border: none;
    height: 100px;
    resize: none;
    font-size: 18px;
    font-family: Arial;
    color: #00896d;
}
```

Kết quả:



3.12 SỬ DỤNG BIẾN

- Trong CSS cũng có thể khai báo các biến và sử dụng lại trong các CSS.
- Các biến trong CSS phải được khai báo trong CSS selectors
- Đối với phạm vi toàn thể, có thể sử dụng :root hoặc body selector
- Tên của biến phải bắt đầu với hai dấu gạch ngang (--).

Ví dụ:

```
:root {
    --primary-color: #1abc9c;
    --primary-dark-color: #00a886;
}
```

Để sử dụng biến, dùng hàm **var()**. Cú pháp hàm **var()** như sau:

```
var(custom-name, value)
```

Trong 2 thuộc tính trên:

- custom-name là bắt buộc. Tên thuộc tính tùy chỉnh (phải bắt đầu bằng hai dấu gạch ngang)
- value là tùy chọn. Giá trị dự phòng (được sử dụng khi thuộc tính tùy chỉnh không hợp lệ)

Ví dụ:

```
input[type=reset] {
    padding: 10px;
    font-size: 18px;
    border: 1px solid var(--primary-color);
    background-color: white;
    color: var(--primary-dark-color);
}`
```

THỰC HÀNH

CÂU HỎI ÔN TẬP LÝ THUYẾT

1. Mã HTML chính xác để tham chiếu đến CSS bên ngoài là gì?

A. <style src="mystyle.css">

B. <stylesheet>mystyle.css</stylesheet>

C. <link rel="stylesheet" type="text/css" href="mystyle.css">
D. <link rel="stylesheet" type="text/css" url="mystyle.css">

2. Vị trí nào trong tài liệu HTML là nơi chính xác để tham chiếu đến CSS bên ngoài?

A. Trong phần <body>
B. Cuối của tài liệu HTML
C. Trong phần <head>
D. Trong phần <footer>

3. Thẻ HTML nào được sử dụng để định nghĩa một CSS trong (Internal CSS)?

A. <script>
B. <style>
C. <css>
D. <internal>

4. Thuộc tính HTML nào được sử dụng để xác định CSS kiểu nội tuyến?

A. style
B. font
C. styles
D. class

5. Đâu là cách để chèn chú thích vào tệp CSS?

A. // this is a comment
B. // this is a comment //
C. /* this is a comment */
D. ' this is a comment

6. Đâu là cách để thêm màu nền cho tất cả các phần tử <h1>?

A. all.h1 {background-color:#FFFFFF;}
B. h1.all {background-color:#FFFFFF;}
C. h1 {background-color:#FFFFFF;}
D. h1: all {background-color:#FFFFFF;}

7. Thuộc tính CSS nào được sử dụng để thay đổi màu văn bản của một phần tử?

A. text-color
B. Color

C. Fgcolor
D. font-color

8. Thuộc tính CSS nào thay đổi kích thước văn bản?

A. font-size
B. text-style
C. text-size
D. font-style

9. Cú pháp CSS chính xác để in đậm tất cả các phần tử <p> là gì?

A. p {font-weight:bold;}
B. <p style="font-size:bold;">
C. p {text-size:bold;}
D. <p style="text-size:bold;">

10. Đâu là cú pháp CSS chính xác?

A. body {color: black;}
B. {body;color:black;}
C. body:color=black;
D. {body:color=black;}

BÀI TẬP TỰ THỰC HÀNH

Bài 1. Sử dụng màu sắc

Tạo một trang web với bố cục đơn giản và sử dụng màu sắc để tạo điểm nhấn. Có thể chọn màu sắc phù hợp với nội dung của trang web và sử dụng các thuộc tính như background-color, color, border-color để thiết lập màu sắc cho các phần tử. Ngoài ra, cũng có thể sử dụng màu sắc gradient để tạo ra hiệu ứng màu sắc trên trang web màu sắc phù hợp với nội dung của trang web.

Bài 2. Tạo và định dạng bảng, văn bản

Tạo một bảng đơn giản và sử dụng CSS để thiết kế các phần tử trên bảng. Có thể sử dụng các thuộc tính CSS như border, background-color, text-align, và padding để định dạng văn bản trên bảng.

Bài 3. Sửa form đã thiết kế

Hãy sửa lại form đã thiết kế trong chương trước để có một giao diện đẹp, hợp lý hơn. Có thể sử dụng các thuộc tính CSS như border-radius, width và height để tạo hình dạng và kích thước phù

hợp cho form. Có thể sử dụng CSS để thêm icon cho các trường nhập liệu trên form. Ví dụ như thêm icon email cho trường nhập email hoặc icon điện thoại cho trường nhập điện thoại.

Bài 4. Tạo form đăng nhập

Tạo một hộp đăng nhập đơn giản với CSS. Có thể thiết kế hộp đăng nhập với các thuộc tính như border, background, và padding. Sau đó, bạn có thể sử dụng thuộc tính hover để thêm hiệu ứng đổi màu khi di chuột qua hộp đăng nhập.

TÀI LIỆU THAM KHẢO

[1] Anne Boehm, Zak Ruvalcaba (2018) Murach's HTML5 and CSS3, Murack.

[2] Responsive Web Design with HTML5 and CSS: Build future-proof responsive Websites using the latest HTML5 and CSS techniques, 3rd, (2020), Ben Frain, Packt Publishing

[3] Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics (2018), Jennifer Robbins, O'Reilly Media

[4] The Book of CSS3, 2nd Edition: A Developer's Guide to the Future of Web Design (2014), Peter Gasston, No Starch Press

[5] Pro CSS3 Layout Techniques (2016), Sam Hampton-Smith, Apress

CHƯƠNG 4: THIẾT KẾ BỐ CỤC

Trong chương này sẽ giới thiệu cách sử dụng CSS để kiểm soát bố cục của một trang Web. Điều đó có nghĩa là người học có thể kiểm soát vị trí của mỗi phần tử HTML xuất hiện trên trang Web. Sau khi học xong chương này, người học có thể triển khai các bố cục mong muốn liên quan đến một trang Web. Ngoài ra, một công nghệ tiên tiến trong thiết kế Web là Web đáp ứng cũng được đề cập nhằm giúp một trang Web có thể hiển thị tối ưu trên các thiết bị có độ rộng màn hình khác nhau.

4.1 ĐỘ RỘNG VÀ CHIỀU CAO

4.1.1 Thuộc tính width và height

Các thuộc tính **width** và **height** có thể là **giá trị số** để xác định chiều rộng, chiều cao tính bằng các đơn vị px, cm,… hoặc các giá trị như sau:

- **auto** - Mặc định. Trình duyệt tính toán chiều cao và chiều rộng
- **%** - Xác định chiều cao / chiều rộng tính theo phần trăm của khối chứa
- **inherit** - Chiều cao /chiều rộng sẽ được kế thừa từ giá trị cha của nó

Ví dụ đặt **chiều cao** và **chiều rộng** của phần tử <**div**>:

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <style>
        div {
            height: 100px;
            width: 50%;
            background-color: #1abc9c;
            text-align: center;
            color: white;
            padding-top: 30px;
        }
    </style>
</head>
<body>
<div>
<div>
    <h1>Tiêu đề h1</h1>
    <h2>Tiêu đề h2</h2>
    <p>Đoạn văn 1</p>
    <p>Đoạn văn 2</p>
</div>
</body>
</html>
```

Các thuộc tính **height** và **width** được sử dụng để đặt chiều cao và chiều rộng của một phần tử. Các thuộc tính chiều cao và chiều rộng không bao gồm phần **padding**, **border** hoặc **margin**.

Hình 4-1 Thiết lập chiều cao và chiều rộng cho một phần tử khối

4.1.2 Chiều rộng và chiều cao tối đa

Chiều rộng và chiều cao tối đa có thể được thiết lập bằng thuộc tính **max-width**, và **max-height.** Ví dụ:

```
div {
    height: 100px;
    max-width: 800px;
    background-color: #1abc9c;
}
```

Trong trường hợp trên nếu thay **max-width** bằng **width** thì kích thước của khối <**div**> sẽ là 800px và trình duyệt sẽ xuất hiện thanh trượt ngang nếu không gian không đủ. Còn nếu sử dụng **max-width**, khối div sẽ tự thu nhỏ kích thước lại nếu không gian không đủ. Xét một ví dụ khác:

```
div {
   width: 80%;
   max-width: 500px;
   height: 100px;
   background-color: #1abc9c;
}
```

Các thuộc tính **max-width** được sử dụng để thiết lập chiều rộng tối đa của một phần tử. Có max-width thể được chỉ định trong các giá trị độ dài, như px, cm, ... hoặc tính bằng phần trăm (%). Trong trường hợp trên, nếu độ rộng của khối **div** được thiết lập chiếm 80% kích thước của phần tử chứa nó, nhưng không được quá 800px.

4.2 ĐƠN VỊ KÍCH THƯỚC

4.2.1 Các loại đơn vị đo kích thước

Có rất nhiều loại đơn vị đo để xác định **kích thước**, chia làm 2 loại chính:

- Đơn vị tuyệt đối
- Đơn vị tương đối

Có rất nhiều thuộc tính như **margin**, **padding**, **font-size**, **border** v.v… sẽ phải sử dụng đến kích thước như **10px**, **2em**… Chú ý khi viết kích thước kèm theo đơn vị trừ trường hợp số 0 thì không cần đơn vị. Đối với một số thuộc tính CSS, giá trị âm có thể được sử dụng. Trong CSS có 2 loại đơn vị đó là đơn vị **tuyệt đối** (absolute) và đơn vị **tương đối** (relative).

4.2.2 Đơn vị tuyệt đối

Đơn vị tuyệt đối là đơn vị chỉ một độ dài cố định, các đơn vị có thể là:

- **cm**: centimeters
- **mm**: millimeters
- **in**: inches (1in = 96px = 2.54cm)
- **px**: pixels (1px = 1/96th of 1in)
- **pt**: points (1pt = 1/72 of 1in)
- **pc**: picas (1pc = 12 pt)

Đơn vị px (Pixel) sẽ liên quan đến thiết bị. Đối với các thiết bị có độ phân giải thấp, 1px sẽ tương đương với một **điểm ảnh** trên màn hình. Tuy nhiên cần chú ý rằng: trừ các thiết bị in ấn cho ra kết quả chính xác, còn đối với thiết bị hiển thị ra màn hình, kích thước theo đơn vị **cm**, mm**, in**, **pt** và **pc** sẽ chỉ là gần đúng. Vì vậy, kết quả có xu hướng khác nhau giữa các thiết bị. Ví dụ khi thiết lập độ dài là 20cm, thì khi chiếu trên màn hình máy tính có thể sẽ không chính xác tuyệt đối là 20cm. Còn khi hiển thị lên các thiết bị như máy chiếu thì chắc chắn kích thước sẽ được phóng to lên rất nhiều. Đây cũng là điều rất dễ hiểu.

4.2.3 Đơn vị tương đối

Một số đơn vị tương đối được sử dụng trong thiết kế Web như sau:

- **em**: Tương đối so với cỡ font hiện tại
- **ex**: Tương đối so với x-height của font hiện tại.
- **ch**: Tương đối so với độ rộng cửa chữ số 0
- **rem**: Tương đối so với phần tử html
- **vw**: Tương đối với độ rộng của viewport (%)
- **vh**: Tương đối với độ cao của viewport (%)
- **vmin**: Tương đối so với kích thước nhỏ hơn của chiều rộng, cao của viewport
- **vmax**: Tương đối so với kích thước lớn hơn của chiều rộng, cao của viewport
- **%**: Có giá trị tương đối so với phần tử cha.

Các đơn vị như em và rem thường được sử dụng để thiết kế những bố cục dễ dàng co giãn mở rộng. Các đơn vị như ex, ch rất ít được sử dụng. Xét ví dụ

```
h1 {
    font-size: 3em;
}
h2 {
```

```
    font-size: 1em;
}
```

Kết quả cho thấy phần tử <**h2**> có kích thước bằng kích thước của chữ bình thường được khai báo trong phần tử Đoạn văn <**p**>

Hình 4-2 Cấu hình cỡ chữ sử dụng đơn vị tương đối

Đơn vị độ dài tương đối nên được ưu tiên sử dụng hơn khi thiết kế Web hoạt động trên nhiều loại thiết bị.

4.3 THUỘC TÍNH MARGIN

4.3.1 Margin là gì

Các thuộc tính **margin** (lề) được sử dụng để tạo không gian xung quanh các phần tử, bên ngoài phần đường viền.

Có 4 thuộc tính Margin xác định khoảng cách cho 4 phía:

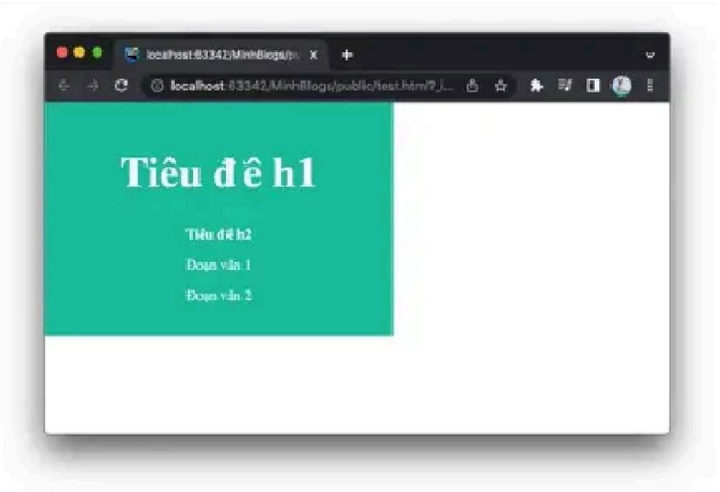
- **margin-top:** khoảng cách bên trên
- **margin-right:** khoảng cách bên phải
- **margin-bottom:** khoảng cách phía dưới
- **margin-left:** khoảng cách bên trái

Tất cả các thuộc tính margin có thể có thể đặt như sau:

- **auto** - trình duyệt tự tính toán lề
- **giá trị** - chỉ định lề trong px, pt, cm, v.v.
- **%** - chỉ định lề theo % chiều rộng của phần tử chứa (container)
- **inherit** - chỉ định rằng lề phải được kế thừa từ phần tử cha

Ví dụ, mặc định phần tử <**body**>sẽ có margin mặc định là 8px, để bỏ phần margin này đi, cần khai báo css cho phần tử <**body**> như sau:

```
body{
    margin: 0px;
}
```



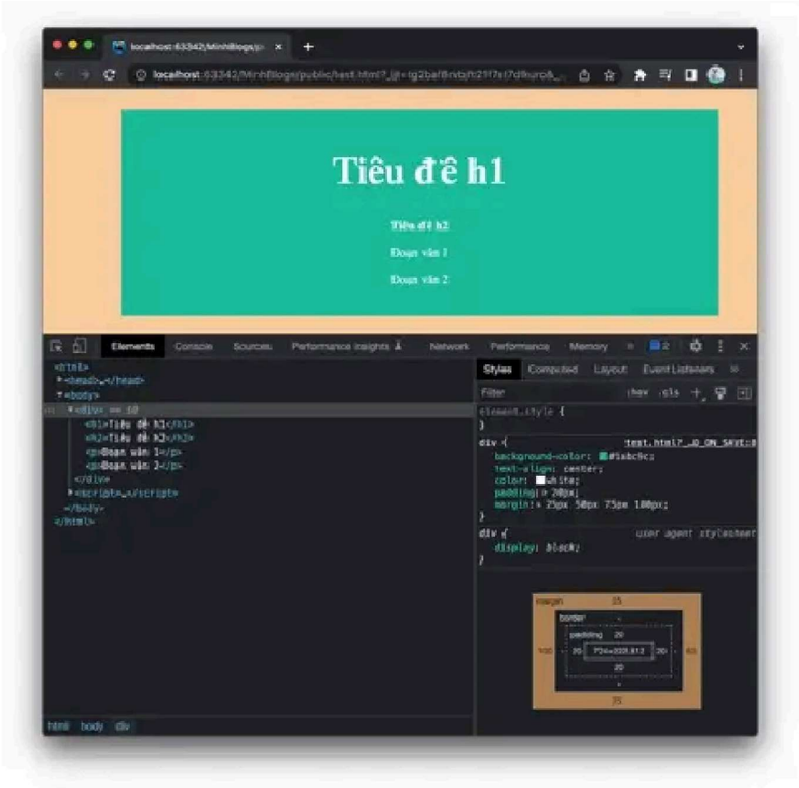
Hình 4-3 Phần tử <body> sau khi bỏ margin

4.3.2 Cách viết rút gọn

Có thể chỉ định tất cả các thuộc tính lề trong một thuộc tính **margin**. Ví dụ lề trên là **25px**, lề phải là **50px**, lề dưới là **75px**, lề trái là **100px**:

```
div {
    background-color: #1abc9c;
    text-align: center;
    color: white;
    padding: 20px;
    margin: 25px 50px 75px 100px;
}
```

Để biết được chính xác margin bằng bao nhiêu, có thể sử dụng đến công cụ **Developer Tools** của trình duyệt, khi đó các thông tin về kích thước, padding, margin được thể hiện rất rõ. Ví dụ công cụ của trình duyệt Chrome:



Hình 4-4 Xem kích thước, padding và margin bằng trình duyệt

Lề trên và dưới là **25px**, Lề trái và phải là **50px**:

```
div {
    background-color: #1abc9c;
    text-align: center;
    color: white;
    padding: 20px;
    margin: 25px 50px;
}
```

Tất cả các lề là **45px**:

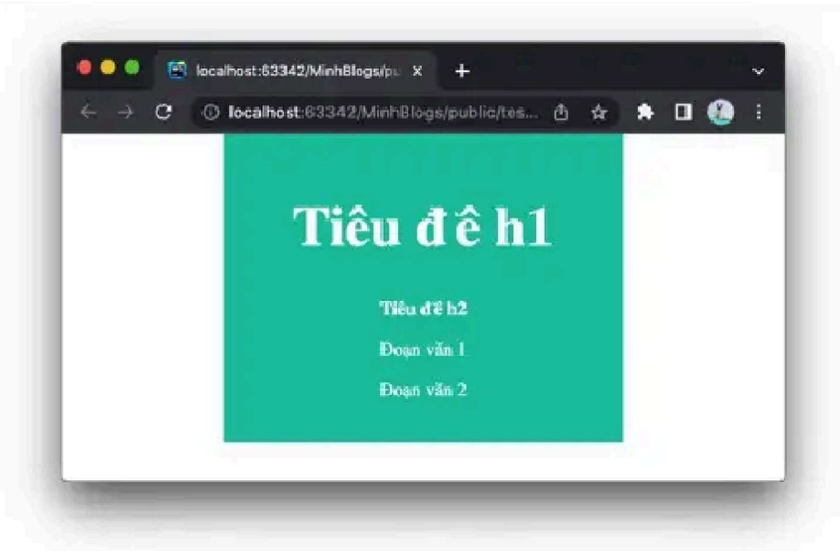
```
div {
    background-color: #1abc9c;
    text-align: center;
    color: white;
    padding: 20px;
    margin: 45px;
}
```

4.3.3 Giá trị auto

Có thể đặt thuộc tính lề là auto để căn giữa một phần tử trong khung chứa. Ví dụ:

```
div {
    width: 300px;
    background-color: #1abc9c;
    text-align: center;
    color: white;
    padding: 20px;
    margin: auto;
}
```

Khi sử dụng thuộc tính **auto**, phần tử này sẽ sử dụng một không gian được quy định trước (như trong trường hợp trên là **300px**, và phần không gian còn lại sẽ được chia đều cho lề trái và phải. Kết quả hiển thị trên trình duyệt sẽ như sau:



Hình 4-5 Sử dụng margin:auto để căn giữa một phần tử khối

4.3.4 Giá trị inherit

Giá trị kế thừa (inherit) sẽ báo rằng margin sẽ tương tự như phần tử cha (chứa nó). Ví dụ:

```
div {
    width: 300px;
    background-color: #1abc9c;
    color: white;
    padding: 20px;
    margin: 30px;
}
p{
    margin: inherit;
    border: 1px solid black;
}
```

Trong đoạn mã trên, phần tử <**p**> kế thừa giá trị thuộc tính margin của phần tử cha (phần tử <**div**>)

Hình 4-6 Phần tử <p> kế thừa giá trị margin từ phần tử cha

4.3.5 Gom lề

Lề trên và dưới của các tử đôi khi được thu gọn thành một lề duy nhất với lề lớn nhất trong hai lề. Xét ví dụ sau:

```
h1 {
    margin: 0 0 50px 0;
}
h2 {
    margin: 30px 0 0 0;
}
```

Trong ví dụ trên, phần tử <**h1**> có lề dưới là 50px và phần tử <**h2**> có lề trên được đặt thành 20px. Thông thường lề dọc giữa <**h1**> và <**h2**> sẽ có tổng cộng 70px (50px + 20px). Nhưng do gom lề, lề thực tế là **50px**. Tuy nhiên, điều này không xảy ra ở lề trái và phải.

4.4 THUỘC TÍNH PADDING

4.4.1 Padding là gì

Thuộc tính **padding** là thuộc tính được sử dụng để tạo không gian xung quanh nội dung của một phần tử, bên trong phần đường viền:

- **padding-top:** khoảng cách trên
- **margin-right:** khoảng cách phải
- **padding-bottom**: khoảng cách dưới
- **padding-left:** khoảng cách trái

Tất cả các thuộc tính lề có thể có các giá trị sau:

- **giá trị** - chỉ định lề trong px, pt, cm, v.v.
- **%** - chỉ định lề theo % chiều rộng của phần tử chứa (container)
- **inherit** - chỉ định rằng lề phải được kế thừa từ phần tử cha

Ví dụ:

```
<!DOCTYPE html>
<html>
<head>
   <style>
       div {
           border: 1px solid #4CAF50;
           padding-top: 100px;
           padding-bottom: 100px;
           padding-right: 150px;
           padding-left: 80px;
       }
   </style>
</head>
<body>

<div>Phần tử Div</div>

</body>
</html>
```

4.4.2 Cách viết rút gọn

Có thể chỉ định tất cả các thuộc tính lề trong một thuộc tính **padding**. Ví dụ padding trên là 25px, **padding** phải là 50px, **padding** dưới là 75px, **padding** trái là 100px:

```
div {
   border: 1px solid #4CAF50;
   padding: 25px 50px 75px 100px;
}
```

padding trên và dưới là 25px, **padding** trái và phải là 50px:

```
div {
   border: 1px solid #4CAF50;
   padding: 25px 50px;
}
```

Như cách viết trên tất cả các **padding** sẽ là 45px:

```
div {
   border: 1px solid #4CAF50;
   padding: 45px;
}
```

4.5 MÔ HÌNH HỘP

Trong CSS, thuật ngữ mô hình hộp hay **Box model** được sử dụng khi nói về thiết kế và bố cục. Nó bao gồm **margin**, **border**, **padding** và **content** (nội dung thực tế). Hình ảnh dưới đây minh họa mô hình hộp:

Giải thích về các phần khác nhau:

- **Content** - Nội dung của hộp, nơi chứa các phần tử khác.
- **Padding** - Không gian xung quanh nội dung
- **Border** - Đường viền bao quanh phần padding và nội dung
- **Margin** - Không gian bên ngoài border

Để đặt chính xác chiều rộng và chiều cao của một phần tử trong tất cả các trình duyệt, cần biết **Box model** hoạt động như thế nào. Xét một ví dụ như sau:

```
<html>
<head>
    <meta charset="UTF-8">
    <style>
       body{
           margin: 0px;
       }
       div {
           width: 320px;
           padding: 20px;
           margin: 20px;
           background-color: #1abc9c;
           color: white;
       }
    </style>
</head>
<body>
<div>
    Phần tử div
</div>
</body>
</html>
```

Trong ví dụ trên kích thước của hộp sẽ bằng: **320px** (độ rộng) + **20px** (padding trái,phải) + **10px** (viền trái, phải) + **0px** (margin trái, phải) = **350px**

Chú ý, thuộc tính **width** chỉ định chiều rộng của vùng nội dung của phần tử. Vùng nội dung là phần bên trong phần **padding**, **border** và **margin** của một phần tử .Vì vậy, nếu một phần tử có chiều rộng được chỉ định, các thiết lập này làm tăng kích thước của phần tử đó. Đây thường là một kết quả không mong muốn. Để giữ chiều rộng **width**, có thể sử thuộc tính **box-sizing** với

giá trị truyền vào là **border-box**. Điều này làm cho phần tử duy trì chiều rộng của nó; nếu tăng phần **padding**, không gian nội dung khả dụng sẽ giảm.

```
div {
    width: 320px;
    padding: 20px;
    margin: 20px;
    background-color: #1abc9c;
    color: white;
    box-sizing: border-box;
}
```

Để sử dụng **box-sizing** cho toàn bộ các phần tử trong tài liệu HTML:

```
* {
    box-sizing: border-box;
}
```

4.6 CĂN CHỈNH

Thuộc tính **align** là sự thiết lập căn chỉnh vị trí của phần tử hoặc nội dung của phần tử. Ví dụ sau, các phần tử không được căn chỉnh, mặc định sẽ căn chỉnh về phía bên trái:

```
<div>
    <h1>Tiêu đề h1</h1>
    <h2>Tiêu đề h2</h2>
    <p>Đoạn văn 1</p>
    <p>Đoạn văn 2</p>
</div>
```

4.6.1 Căn giữa cho phần tử

Để căn giữa một phần tử khối (như <**div**>), ta sử dụng thuộc tính **margin** có giá trị là **auto** để căn chỉnh phần tử nằm ở giữa so với phần tử cha của nó. Phần tử sau khi chiếm không gian nhất định, khoảng trống còn lại sẽ được chia đều cho hai bên lề. Ví dụ:

```
div {
    width: 320px;
    padding: 20px;
    background-color: #1abc9c;
    color: white;
    margin: auto;
}
```

Lưu ý:

- Căn chỉnh giữa sẽ không có tác dụng nếu không có **width** (hoặc width có giá trị là 100%)
- Thuộc tính **margin**: auto chỉ căn giữa cho phần tử block như <**div**>, <**p**>, <**h1**> đến <**h6**>, <**header**>, <**footer**>, <**section**>, <**nav**>... còn các phần tử inline như <**span**>, <**a**>, <**strong**>, <**b**>, <**i**>... sẽ không được áp dụng.
- Có thể ghi đè lại phần tử hiển thị block hay inline bằng cách sử dụng thuộc tính **display**. Ví dụ: span { **display: block;**}.

- Để căn giữa một ảnh, đặt **margin** thành **auto** và làm cho hình ảnh trở thành phần tử dạng **block** như đã nói ở lưu ý phía trên.

4.6.2 Căn chỉnh văn bản

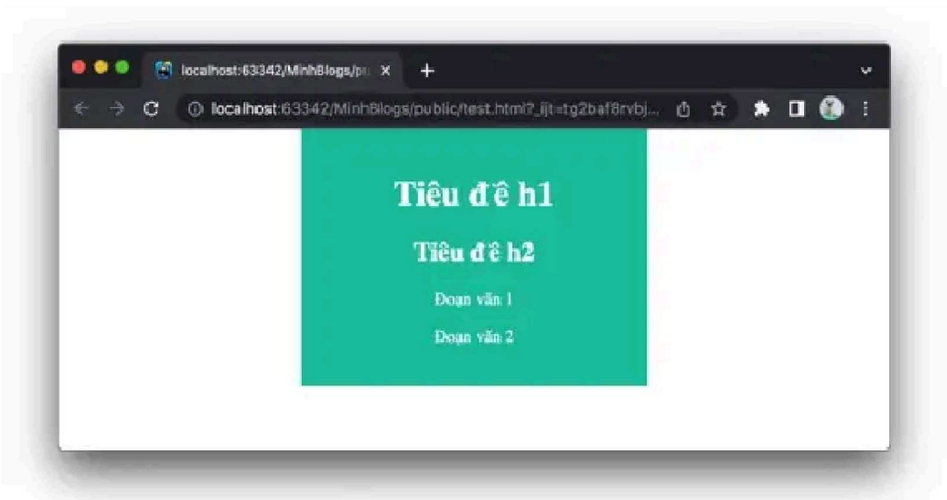
Trong CSS có thuộc tính **text-align** cho phép căn chỉnh văn bản. Nội dung văn bản sẽ nằm bên trái, ở giữa hay bên phải của phần tử bằng các giá trị:

- **center** - căn giữa
- **left** - căn trái
- **right** - căn phải

Ví dụ:

```
div {
    width: 320px;
    padding: 20px;
    background-color: #1abc9c;
    color: white;
    margin: auto;
    text-align: center;
}
```

Lưu ý thuộc tính **text-align** cũng chỉ áp dụng cho phần tử block. Kết quả sau khi căn chỉnh giữa:



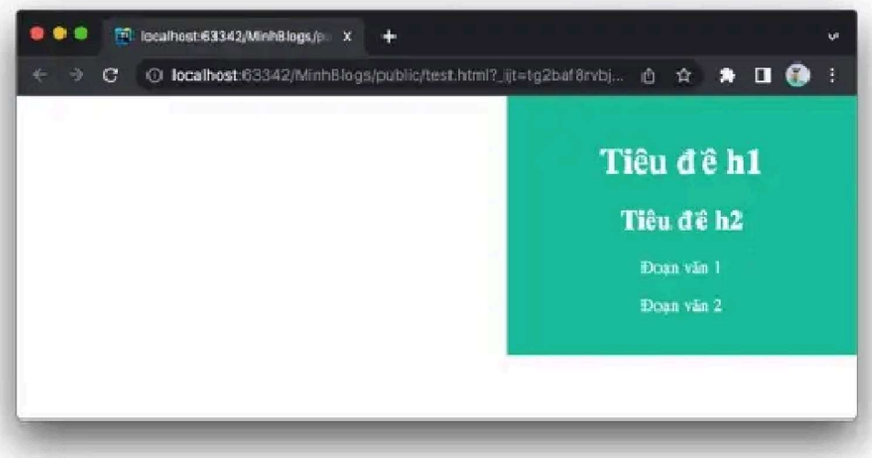
Hình 4-7 Căn chỉnh giữa cho phần tử khối

4.6.3 Căn trái phải cho phần tử

Để căn trái phải cho một phần tử có nhiều cách, ví dụ sử dụng **float**.

```
div {
    width: 320px;
    padding: 20px;
    background-color: #1abc9c;
```

```
    color: white;
    float: right;
    text-align: center;
}
```



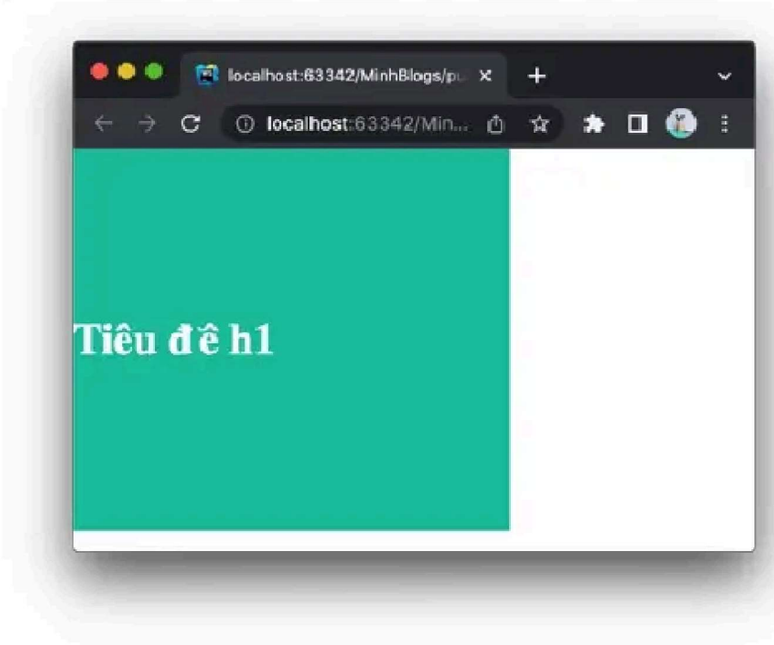
Hình 4-8 Sử dụng float để căn phải một phần tử

4.6.4 Căn giữa theo chiều dọc

Có nhiều cách để **căn giữa** một phần tử theo **chiều dọc** trong CSS. Giải pháp đơn giản nhất đó là sử dụng **padding**.

```
div {
    width: 320px;
    padding: 100px;
    background-color: #1abc9c;
    color: white;
}
```

Kết quả:



Hình 4-9 Căn giữa theo chiều dọc

Cách khác để căn giữa theo chiều dọc là sử dụng thuộc tính **line-height** với giá trị giống như thuộc tính **height.**

```
div {
    width: 320px;
    line-height: 200px;
    background-color: #1abc9c;
    color: white;
}
```

Ngoài ra cũng có thể kết hợp sử dung 2 thuộc tính **position** và **transform**. Ví dụ:

```
div {
    width: 320px;
    position: relative;
    background-color: #1abc9c;
    color: white;
    height: 300px;
}
h1{
    position: absolute;
    top: 50%;
    left: 50%;
    margin: 0px;
    transform: translate(-50%, -50%);
}
```

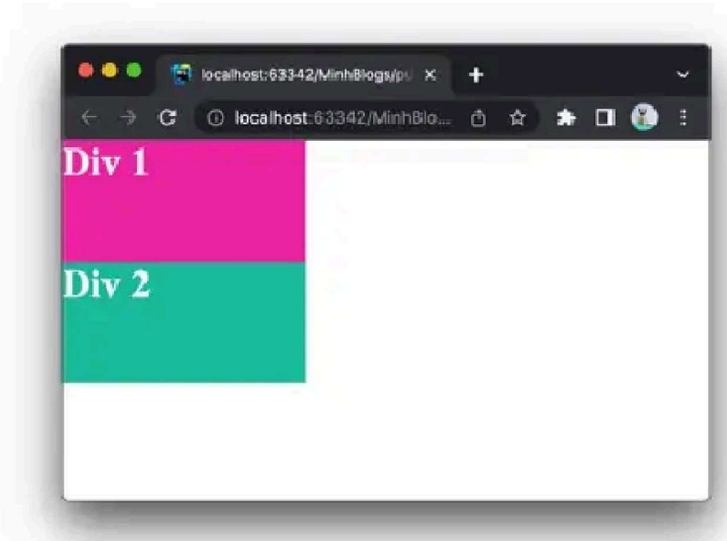
4.7 THIẾT LẬP VỊ TRÍ

Thuộc tính **position** xác định loại của phương pháp định vị trí cho thành phần. Thuộc tính **position** có các loại sau đây:

- static
- relative
- fixed
- absolute
- sticky

Ví dụ xét một thẻ div:

```
<html>
<head>
    <meta charset="UTF-8">
    <style>
       * {
           box-sizing: border-box;
           margin: 0;
       }
       div{
           width: 200px;
           height: 100px;
           color: white;
       }
       div.div1{
           background-color: #ed25a4;
       }
       div.div2{
           background-color: #1abc9c;
       }

    </style>
</head>
<body>
<div class="div1">
    <h1>Div 1</h1>
</div>
<div class="div2">
    <h1>Div 2</h1>
</div>
</body>
</html>
```



Hình 4-10 Hai phần tử nằm liền kề nhau

4.7.1 static

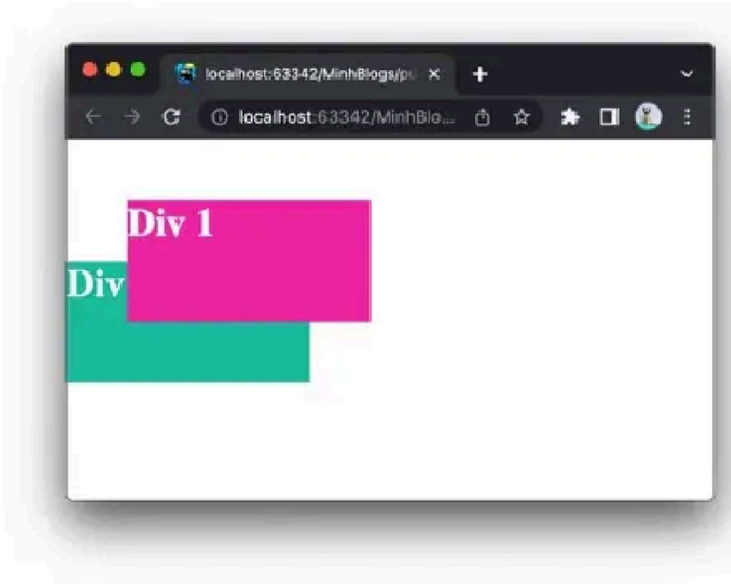
Position thiết lập là **static** là mặc định. Với giá trị này vị trí các phần tử sẽ không bị ảnh hưởng bởi các thuộc tính **left**, **right**, **bottom**, **top**.

4.7.2 relative

Khi **position** được thiết lập là **relative**. Vị trí của các phần tử sẽ phụ thuộc tương đối vào vị trí ban đầu của nó. Ví dụ:

```
div.div1{
    background-color: #ed25a4;
    position: relative;
    top: 50px;
    left: 50px;
}
```

Kết quả:



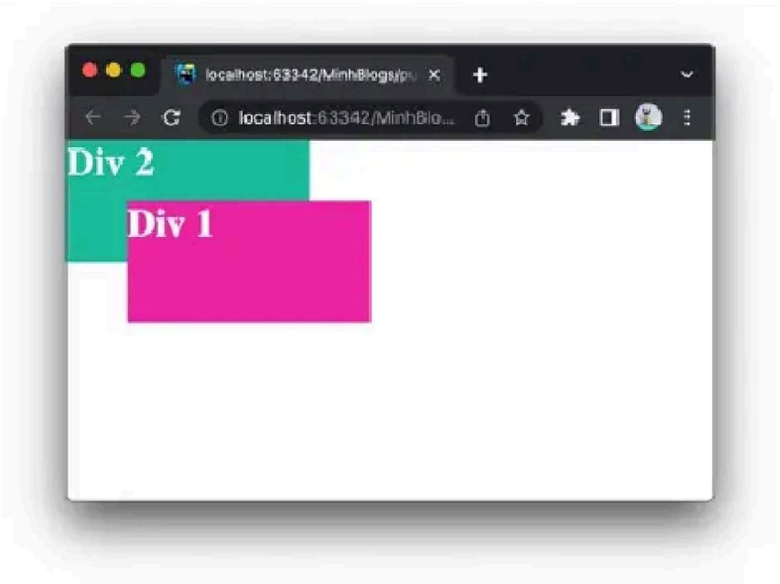
Hình 4-11 Div1 được thiết lập position:relative

4.7.3 fixed

Khi **position** được thiết lập là **fixed**. Vị trí sẽ phụ thuộc tương đối vào cửa sổ trình duyệt. Ví dụ:

```
div.div1 {
    background-color: #ed25a4;
    position: fixed;
    top: 50px;
    left: 50px;
}
```

Kết quả hiển thị:



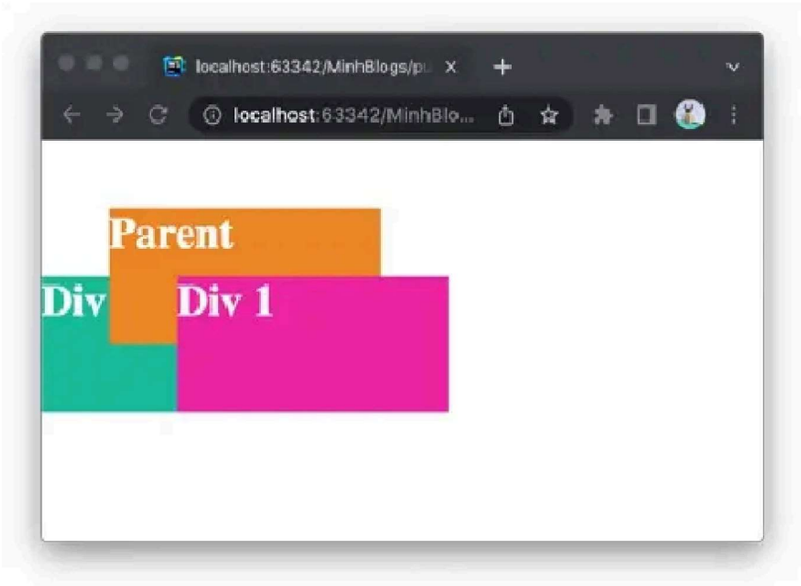
Hình 4-12 Div1 được thiết lập position:fixed

Có thể thấy khi **position** là **fixed**, nó sẽ không chiếm không gian, khiến **div2** bị đẩy lên phía trên.

4.7.4 absolute

Vị trí sẽ phụ thuộc tương đối vào phần tử cha. Ví dụ **div1** nằm 1 trong 1 phần tử **parent**:

```
div.parent {
    background-color: #ed8625;
    position: relative;
    top: 50px;
    left: 50px;
}
div.div1 {
    background-color: #ed25a4;
    position: absolute;
    top: 50px;
    left: 50px;
}
```

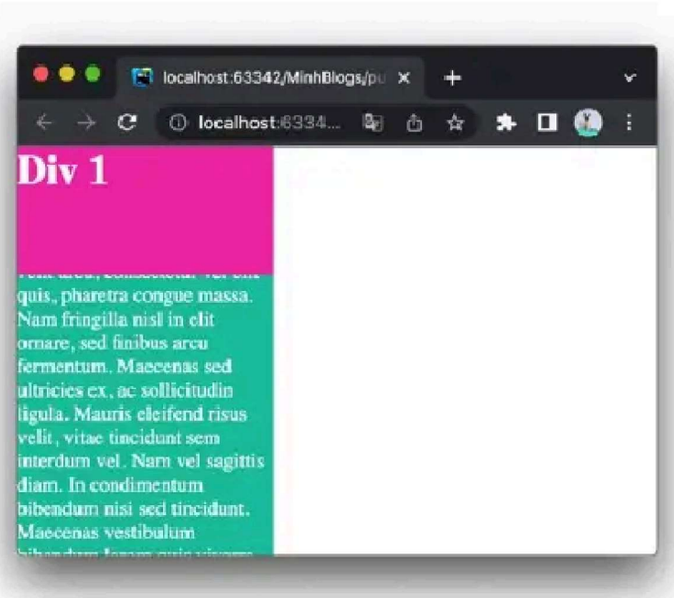


Hình 4-13 Div1 được thiết lập position:absolute

4.7.5 sticky

Vị trí sẽ phụ thuộc vào việc cuộn con trỏ chuột. Ví dụ:

```
* {
    box-sizing: border-box;
    margin: 0;
}

div {
    width: 200px;
    height: 100px;
    color: white;
}

div.div1 {
    background-color: #ed25a4;
    position: sticky;
    top: 0;
}

div.div2 {
    background-color: #1abc9c;
    width: 200px;
    height: 1000px;
}
```



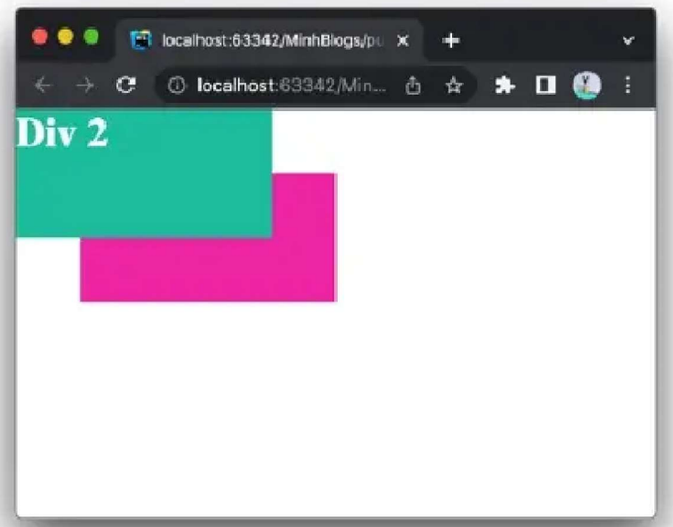
Hình 4-14 Div1 được thiết lập position:fixx

4.7.6 z-index

Thuộc tính **z-index** cho phép thay đổi thứ tự hiển thị theo chiều sâu. Ví dụ:

```
div.div1 {
    background-color: #ed25a4;
    position: sticky;
    top: 0;
    z-index: -1;
}
```

Kết quả sẽ nhận thấy phần tử có **z-index** nhỏ hơn sẽ nằm phía dưới:

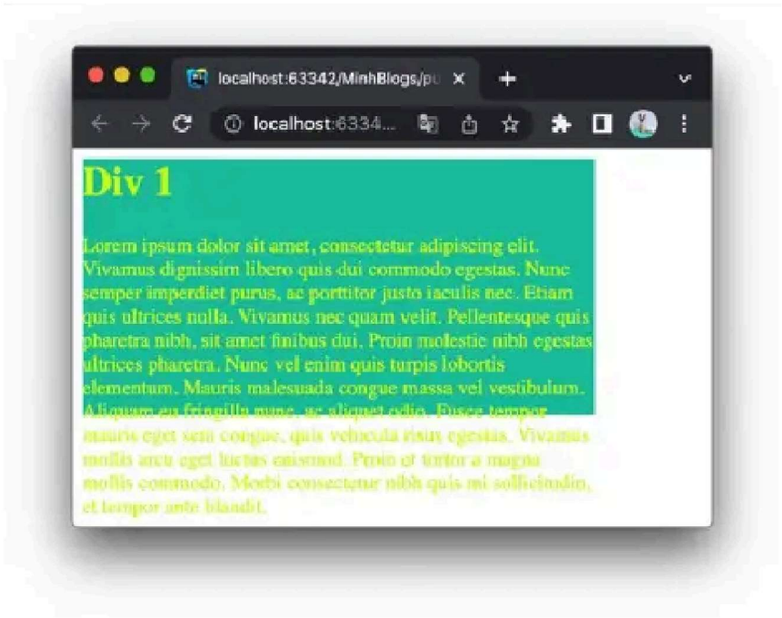


Hình 4-15 Div nằm dưới vì có z-index nhỏ hơn

4.8 THUỘC TÍNH TRÀN OVERFLOW

Thuộc tính **overflow** xác định điều gì sẽ xảy ra nếu một thành phần box tràn nội dung. **overflow** có thể nhận các giá trị:

- visible (mặc định)
- hidden
- scroll
- auto



Hình 4-16 Nội dung bị tràn do dữ liệu bên trong cần nhiều không gian để hiển thị

Giá trị khi không thiết lập overflow hoặc thiết lập **overflow: visible** có nghĩa nội dung dung sẽ **không bị cắt** khi phần nội dung bị tràn ra khỏi box.

Thiết lập **overflow**:**hidden** có nghĩa nội dung sẽ bị ẩn khi phần nội dung bị tràn ra khỏi box:

```
div{
    background-color: #1abc9c;
    width: 400px;
    height: 200px;
    color: #d0ff00;
    overflow: hidden;
}
```

Thiết lập **overflow**:**scroll** có nghĩa nội dung sẽ bị ẩn khi phần nội dung tràn ra khỏi box, tuy nhiên xuất hiện thêm thanh cuộn để có thể cuộn và nhìn thấy nội dung ẩn:

```
div{
    background-color: #1abc9c;
    width: 400px;
    height: 200px;
    color: #d0ff00;
    overflow: scroll;
}
```

Thiết lập **overflow:auto** tương đương với giá trị scroll nhưng thanh cuộn sẽ chỉ xuất hiện trong trường hợp cần thiết:

```
div{
    background-color: #1abc9c;
    width: 400px;
    height: 200px;
    color: #d0ff00;
```

```
    overflow: auto;
}
```

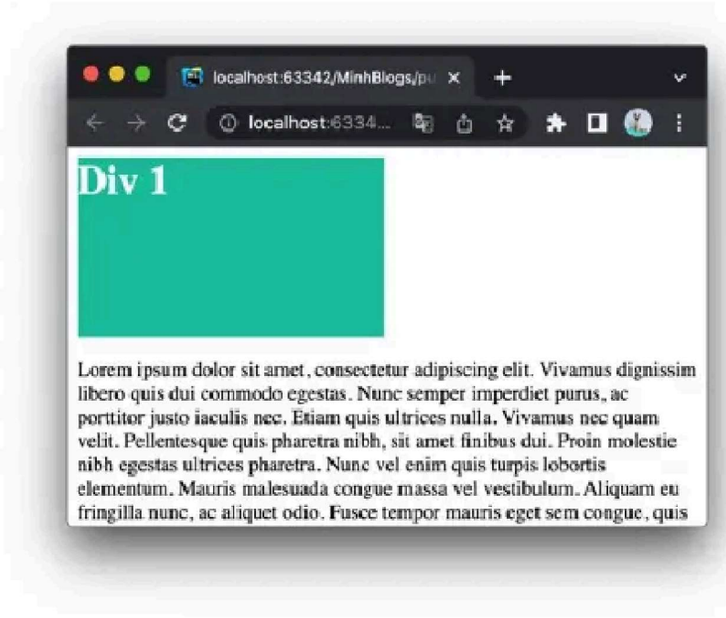
Ngoài các giá trị trên, còn có thể sử dụng các giá trị **overflow-x** và **overflow-y** để xác định những thay đổi khi nội dung tràn theo chiều x và chiều y. Ví dụ khi tràn theo chiều x thì **overflow-x**:**hidden** nội dung còn tràn theo chiều y thì **overflow-x**:**scroll.**

4.9 THUỘC TÍNH TRÔI FLOAT

Thuộc tính **float** xác định có hay không một thành **phần tử khối** được **float** (trôi), và trôi về hướng nào và nằm cạnh nhau thay vì nằm phía trên nhau. Ví dụ:

```
<html>
<head>
    <meta charset="UTF-8">
    <style>
        div{
            background-color: #1abc9c;
            width: 240px;
            height: 140px;
            color: white;
        }
    </style>
</head>
<body>
<div class="div1">
    <h1>Div 1</h1>
</div>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus dignissim
libero quis dui commodo egestas. Nunc semper imperdiet purus, ac porttitor justo
iaculis nec. Etiam quis ultrices nulla. Vivamus nec quam velit. Pellentesque
quis pharetra nibh, sit amet finibus dui. Proin molestie nibh egestas ultrices
pharetra. Nunc vel enim quis turpis lobortis elementum. Mauris malesuada congue
massa vel vestibulum. Aliquam eu fringilla nunc, ac aliquet odio. Fusce tempor
mauris eget sem congue, quis vehicula risus egestas. Vivamus mollis arcu eget
luctus euismod.</p>
</body>
</html>
```

Kết quả hiển thị ví dụ trên:

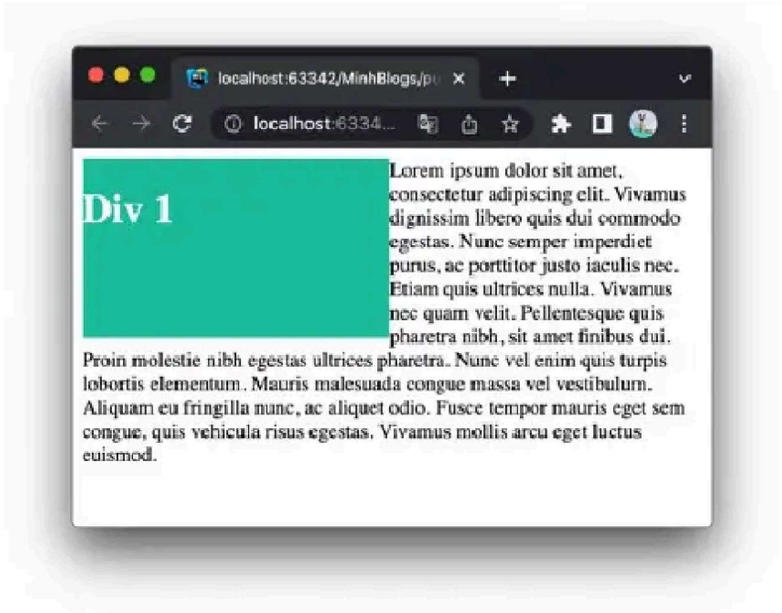


Hình 4-17 Trường hợp div không sử dụng float

Có thể thấy cả phần tử <**div**> và phần tử <**p**> trong ví dụ trên đều là phần tử khối vì vậy sẽ đều chiếm toàn bộ độ rộng của trang Web. Đây cũng giống như khi thiết lập **float:none**. Giá trị **float**: **left** xác định phần tử sẽ trôi về phía **bên trái:**

```
div{
    background-color: #1abc9c;
    width: 240px;
    height: 140px;
    color: white;
    float:left
    margin-right: 20px;
}
```

Kết quả hiển thị:



Hình 4-18 Trường hợp div được thiết lập float:left

Giá trị **right** xác định đối tượng sẽ trôi về phía **bên phải.** Ví dụ:

```
div{
    background-color: #1abc9c;
    width: 240px;
    height: 140px;
    color: white;
    float: right;
    margin-left: 20px;
}
```

Khi sử dụng **float**, các thuộc tính sẽ nằm bên cạnh nếu như không gian vẫn còn, trong trường hợp không muốn điều đó xảy ra, có thể sử dụng thuộc tính **clear**.

Thuộc tính **clear** có thể nhận các giá trị sau:

- **none:** đối tượng sẽ không bị đẩy xuống dưới một đối tượng đang float right hoặc left.
- **left:** đối tượng sẽ đặt dưới một đối tượng đang float left.
- **right**: đối tượng sẽ đặt dưới một đối tượng đang float right.
- **both:** đối tượng sẽ đặt dưới một đối tượng đang float right hoặc float left.
- **inherit:** đối tượng sẽ lấy giá trị clear của phần tử cha.

Ví dụ:

```
<html>
<head>
    <meta charset="UTF-8">
    <style>
        div{
            background-color: #1abc9c;
            width: 240px;
```

```
            height: 140px;
            color: white;
            float: left;
            margin-right: 20px;
        }
        p{
            clear: left;
        }

    </style>
</head>
<body>
<div class="div1">
    <h1>Div 1</h1>
</div>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus dignissim
libero quis dui commodo egestas. Nunc semper imperdiet purus, ac porttitor justo
iaculis nec. Etiam quis ultrices nulla. Vivamus nec quam velit. Pellentesque
quis pharetra nibh, sit amet finibus dui. Proin molestie nibh egestas ultrices
pharetra. Nunc vel enim quis turpis lobortis elementum. Mauris malesuada congue
massa vel vestibulum. Aliquam eu fringilla nunc, ac aliquet odio. Fusce tempor
mauris eget sem congue, quis vehicula risus egestas. Vivamus mollis arcu eget
luctus euismod.</p>
</body>
</html>
```

Xét trường hợp trên, <**p**> sẽ luôn nằm cùng hàng với **div1** cho dù có hay không thuộc tính **float**, để <**p**> nằm dưới, có thể sử dụng thuộc tính **clear**. Ví dụ:

```
p{
    clear: left;
}
```

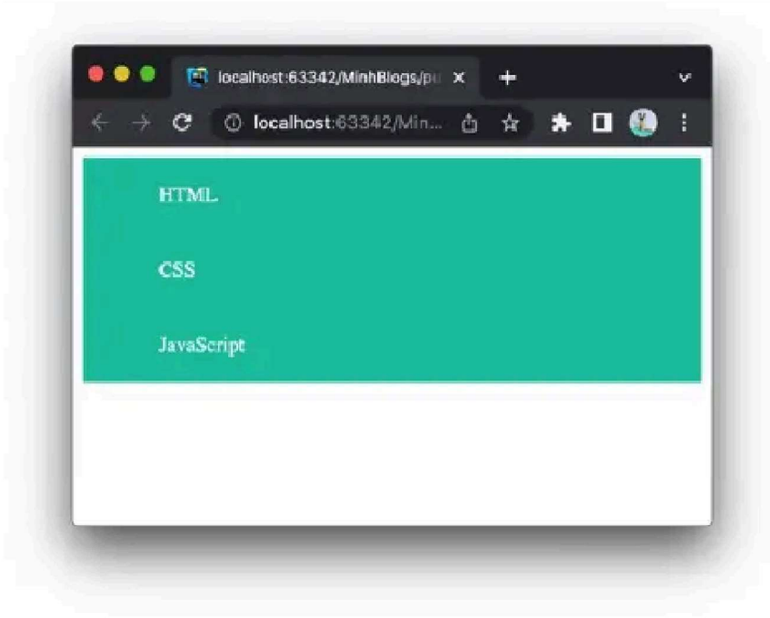
4.10 THUỘC TÍNH DISPLAY

Có thể nhận thấy rằng, một số thẻ HTML như <**div**>, <**p**>, <**ul**> luôn bắt đầu bằng một dòng mới và có độ dài dòng trải dài 100% độ rộng trang Web, trong khi các thẻ HTML khác như <**span**>, <**img**> hoặc <**a**> thì thường được đặt nối tiếp nhau trên cùng một dòng. Điều này do các thẻ mặc định sẽ có 2 loại hiển thị là dạng **khối** (**block**) hay trên **dòng** (**inline**). Thuộc tính **display** có thể thay đổi giá trị mặc định này và khai báo thêm nhiều kiểu định dạng khác. Ví dụ:

```
<html>
<head>
    <style>
        ul{
            background-color: #1abc9c;
            list-style-type:none
        }
        li {
            color: white;
            padding: 20px;
        }
```

```
    </style>
</head>
<body>
<ul>
    <li>HTML</li>
    <li>CSS</li>
    <li>JavaScript </li>
</ul>
</body>
</html>
```

Kết quả khi chạy trên trình duyệt:



Hình 4-19 Các phần tử <li> với thiết lập display mặc định

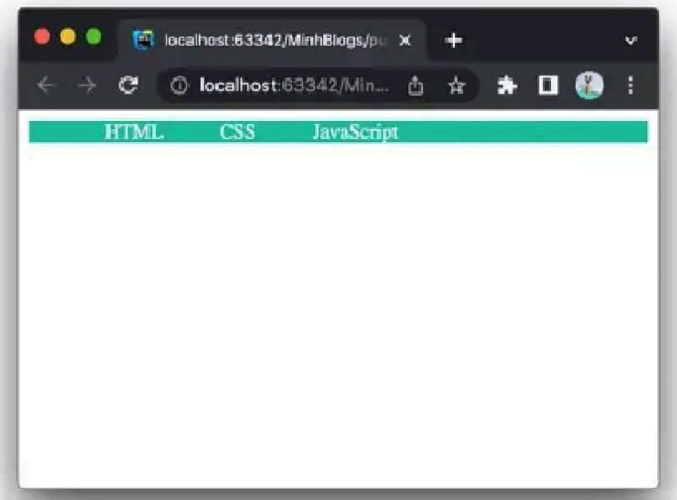
Thuộc tính **display** xác định loại hiển thị của các phần tử **thường** nhận các giá trị như sau:

- **none**: làm ẩn phần tử, nhưng không xóa phần tử này.
- **inline:** là cách hiển thị trên một dòng và chiều rộng của thẻ đó sẽ phụ thuộc vào nội dung bên trong của thẻ (phần tử nội tuyến).
- **block:** là cách hiển thị phần tử luôn bắt đầu trên một dòng mới và có chiều rộng bằng 100% chiều rộng trang Web (phần tử khối).
- **inline-block** là cách hiển thị kết hợp cả hai cách trên, chuyển phần tử về hiển thị trên cùng một hàng nhưng nó vẫn thừa hưởng các đặc tính của block. Ví dụ như có thể thiết lập chiều cao cho khối.

Ví dụ trong trường hợp trên phần tử <**li**> mặc định sẽ có thiết lập **display**:**block**, có thiết lập thành **inline** như sau:

```
li {
    color: white;
    padding: 20px;
```

```
    display: inline;
}
```



Hình 4-20 Các phần tử <li> với thiết lập display:inline

Tuy nhiên có thể thấy rằng nếu <**li**> là một phần tử inline thì sẽ không thể đặt được **padding.** Vì vậy có thể đổi lại như sau:

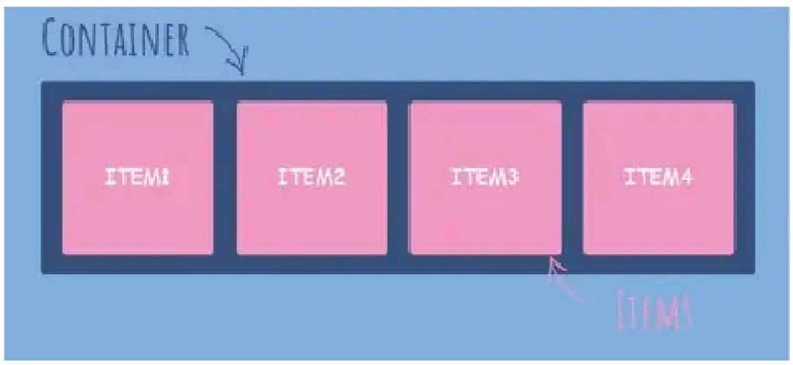
```
li {
    color: white;
    padding: 20px;
    display: inline-block;
}
```

Ngoài **display** nhận giá trị **none**, **block**, **inline**, **inline-block**, nó có thể nhận cá giá trị như flex, hoặc grid nhằm tạo ra các bố cục thiết kế linh hoạt. Các thuộc tính này sẽ được trình bày cụ thể hơn trong phần sau.

4.11 HỘP LINH HOẠT (FLEXBOX)

4.11.1 Flexbox tạo ra các bố cục linh hoạt

flex là một trong những giá trị của thể đặt cho thuộc tính **display**. Giá trị **flex** giúp tạo ra kỹ thuật thiết kế bố cục hộp linh hoạt (Flexbox). **Flexbox** bao gồm việc thiết lập cấu hình cho **Container** và các **Item.** Trong đó Container là các hộp chứa và item là các phần tử bên trong.



Hình 4-21 Hoạt động của flexbox dựa trên việc cấu hình Container và Item

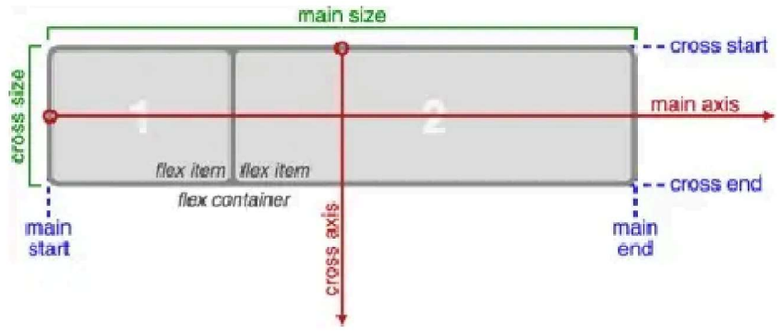
4.11.2 Hộp chứa (flex container)

Để bắt đầu sử dụng các mô hình **Flexbox**, đầu tiên cần phải xác định một **flex container**. Ví dụ tạo **container** cùng các **item** bên trong:

```
<html>
<head>
    <style>
        body{
            font-family: Arial;
        }
        .flex-container {
            display: flex;
            background-color: #1abc9c;
            flex-direction: column;
        }
        .flex-container > div {
            background-color: white;
            margin: 10px;
            padding: 20px;
            font-size: 30px;
        }
    </style>
</head>
<body>
<div class="flex-container">
    <div>1</div>
    <div>2</div>
    <div>3</div>
    <div>4</div>
    <div>5</div>
    <div>6</div>
</div>
</body>
</html>
```

Việc sử dụng flexbox cần nắm rõ một số khái niệm thuật ngữ sau:

- **Các trục**: main axis, cross axis
- **Kích thước**: main size, cross size
- **Các điểm**: main start, main end, cross start, cross end



Hình 4-22 Một số thuật ngữ cần biết khi làm việc với Flexbox

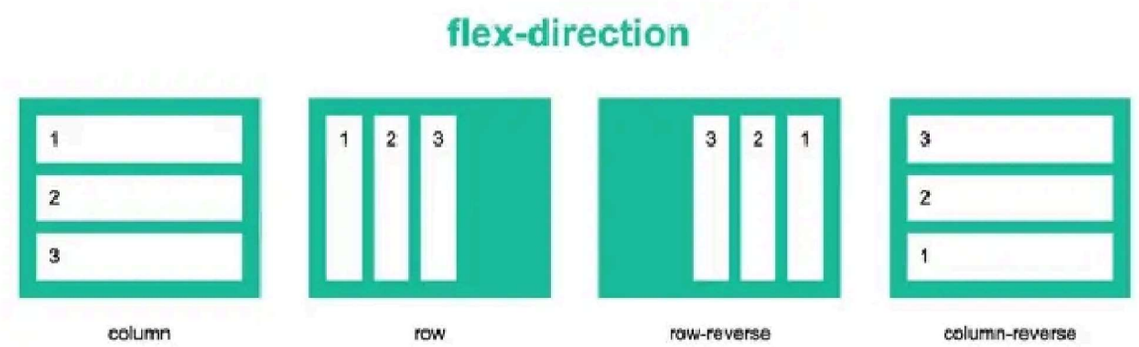
Các thuộc tính có thể thiết lập cho **container**:

- flex-direction
- flex-wrap
- flex-flow
- align-items
- justify-content
- align-content

Thuộc tính **flex-direction** xác định **hướng** mà **container** muốn xếp các **flex items** (chính là hướng của m**ain-axis**). Các giá trị có thể nhận:

- **row (mặc định):** Các phần tử xếp trên 1 hàng.
- **column**: Các phần tử xếp trên 1 cột.
- **row-reverse**: Các phần tử xếp trên một hàng theo chiều từ phải sang trái.
- **column-reverse**: Các phần tử xếp trên một cột theo chiều từ dưới lên trên.

```
.flex-container {
   display: flex;
   background-color: #1abc9c;
   flex-direction: column;
}
```



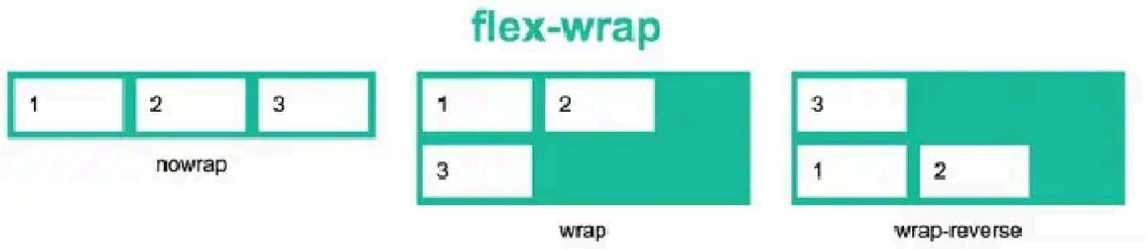
Hình 4-23 Thuộc tính flex-direction

Thuộc tính **flex-wrap** dùng để kiểm soát việc các items sẽ hiển thị như nào khi kích thước theo **main-axis** không đủ. Thuộc tính flex-wrap có thể nhận các giá trị:

- **nowrap (mặc định)**: Các item sẽ trên 1 hàng, cột duy nhất.
- **wrap**: Các items sẽ xuất hiện trên nhiều hàng, cột.
- **wrap-reverse**: Giống wrap nhưng thứ tự các hàng, cột bị đảo.

Ví dụ:

```
.flex-container {
   display: flex;
   background-color: Tomato;
   flex-wrap: wrap;
}
```



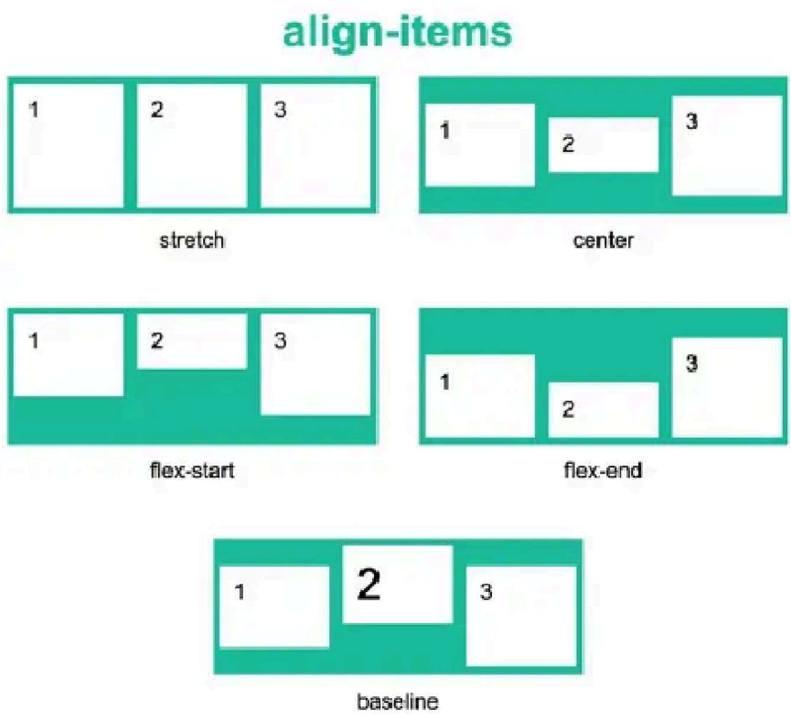
Hình 4-24 Thuộc tính flex-wrap

Thuộc tính **flex-flow** là một thuộc tính **viết tắt (shorthand)** để thiết lập cả hai thuộc tính flex-direction và flex-wrap.

```
.flex-container {
   display: flex;
   background-color: Tomato;
   flex-flow: column nowrap;
}
```

Thuộc tính **align-items** được sử dụng **co giãn**, căn chỉnh các **items** theo **cross-axis**. Các giá trị có thể thiết lập bao gồm:

- **stretch (mặc định):** Các phần tử sẽ được giãn 100% theo trục cross.
- **center:** các phần tử nằm giữa.
- **flext-start:** Các phần tử ở vị trí cross-start.
- **flex-end:** Các phần tử ở vị trí cross-end.
- **baseline:** Các phần tử sẽ bằng nhau phần chân chữ.



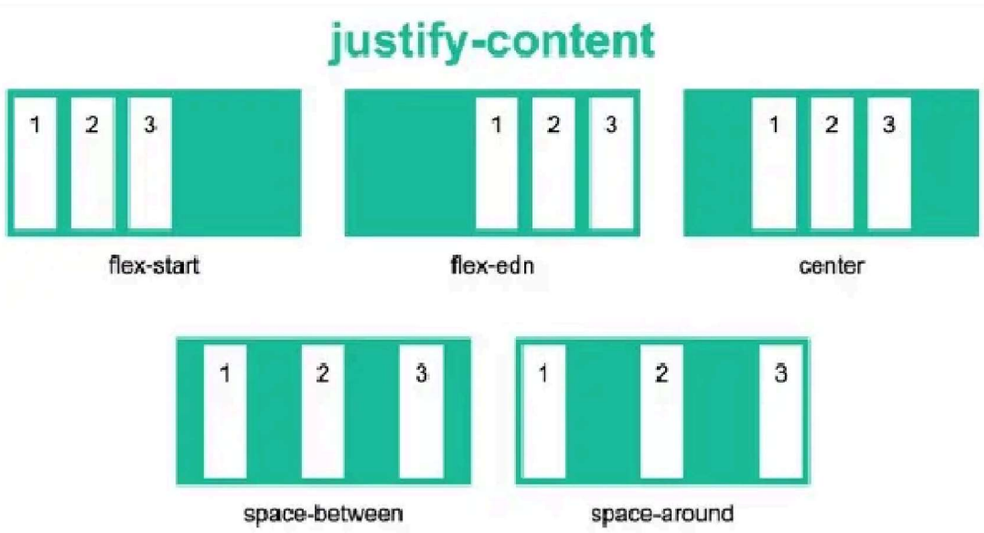
Hình 4-25 Thuộc tính align-items

Ví dụ, khi muốn căn giữa các phần tử khi hiển thị dưới dạng hàng (row):

```
.flex-container {
   display: flex;
   background-color: Tomato;
   align-items: center;
   flex-direction: row;
   height: 200px;
}
```

Thuộc tính **justify-content** được sử dụng để sắp xếp các phần tử theo **main-axis** khi còn thừa không gian. Các giá trị có thể thiết lập bao gồm:

- **flex-start (mặc định):** dồn nội dung sang vị trí main-start.
- **flex-end:** dồn nội dung sang vị trí main-end.
- **center**: căn giữa.
- **space-around:** khoảng cách giữa các item bằng nhau.
- **space-between:** không gian thừa cho mỗi item bằng nhau.



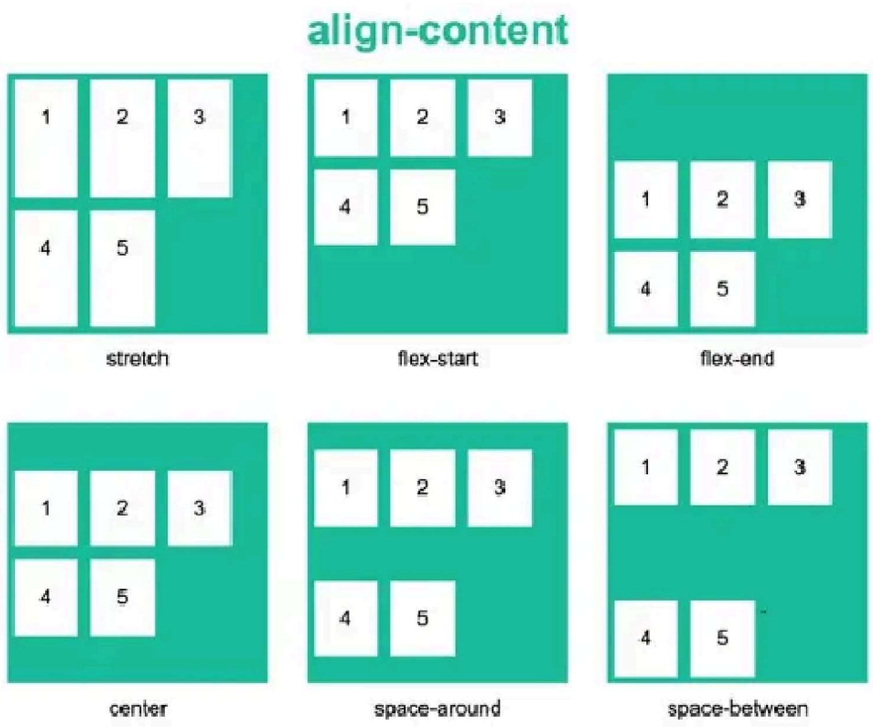
Hình 4-26 Thuộc tính justify-content

Ví dụ:

```
.flex-container {
   display: flex;
   background-color: Tomato;
   justify-content: flex-end;
}
```

Thuộc tính **align-content** được sử dụng để sắp xếp các dòng (lines) theo **cross-axis** khi còn thừa không gian. Các giá trị có thể thiết lập bao gồm:

- **stretch** (mặc định): mặc định.
- **flex-end:** dồn sang vị trí cross-end.
- **flex-start**: dồn sang vị trí cross-start.
- **center**: dồn vào giữa.
- **space-around**: khoảng cách giữa các item bằng nhau.
- **space-between**: không gian thừa cho mỗi item bằng nhau.



Hình 4-27 Thuộc tính align-content

Ví dụ căn để các hàng nằm giữa container:

```
.flex-container {
   display: flex;
   background-color: Tomato;
   height: 300px;
   flex-wrap: wrap;
   align-content: center;
}
```

Ví dụ căn để các cột nằm bên phải container:

```
.flex-container {
   display: flex;
   background-color: Tomato;
   height: 600px;
   flex-direction: column;
   flex-wrap: wrap;
   align-content: flex-end;
}
```

4.11.3 Các phần tử flex item

Trong container có các phần tử con (item), các phần tử đó có thể thiết lập một số các thuộc tính như sau:

- **order:** dùng để xác định thứ tự của các phần tử.
- **flex-grow:** dùng để phát triển (grow) các phần tử.

- **flex-shrink:** dùng để co (shink) các phần tử.
- **flex-basis:** Dùng để xác định kích thước các phần tử.
- **flex:** thuộc tính dùng để viết tắt cho các thuộc tính trên.
- **align-self:** Căn items theo chiều cross-axis.

Thuộc tính **order** xác định thứ tự của các phần tử. Ví dụ:

```
<div class="flex-container">
   <div style="order: 3">1</div>
   <div style="order: 2">2</div>
   <div style="order: 4">3</div>
   <div style="order: 1">4</div>
</div>
```

Thuộc tính **flex-grow** xác định các item sẽ phát triển (grow) tỷ lệ so với các item khác như nào khi thừa không gian. Chú ý giá trị phải là số (mặc định là 0)

```
<div class="flex-container">
   <div style="flex-grow: 1">1</div>
   <div style="flex-grow: 1">2</div>
   <div style="flex-grow: 8">3</div>
</div>
```

Thuộc tính **flex-shink** xác định các item sẽ co (shrink) tỷ lệ so với các item khác như nào khi không có đủ không gian. Chú ý giá trị phải là số (mặc định là 1). Ví dụ khi các phần tử div có width là 100px, và không có đủ không gian, nếu muốn phần tử không shink, có thể đặt như sau:

```
<div class="flex-container">
   <div>1</div>
   <div>2</div>
   <div style="flex-shrink: 0">3</div>
   <div>4</div>
   <div>5</div>
   <div>6</div>
   <div>7</div>
   <div>8</div>
   <div>9</div>
   <div>10</div>
</div>
```

Thuộc tính **flex-basis** xác định kích thước của 1 item theo chiều trục main. Ví dụ:

```
<div class="flex-container">
   <div>1</div>
   <div>2</div>
   <div style="flex-basis: 200px">3</div>
   <div>4</div>
</div>
```

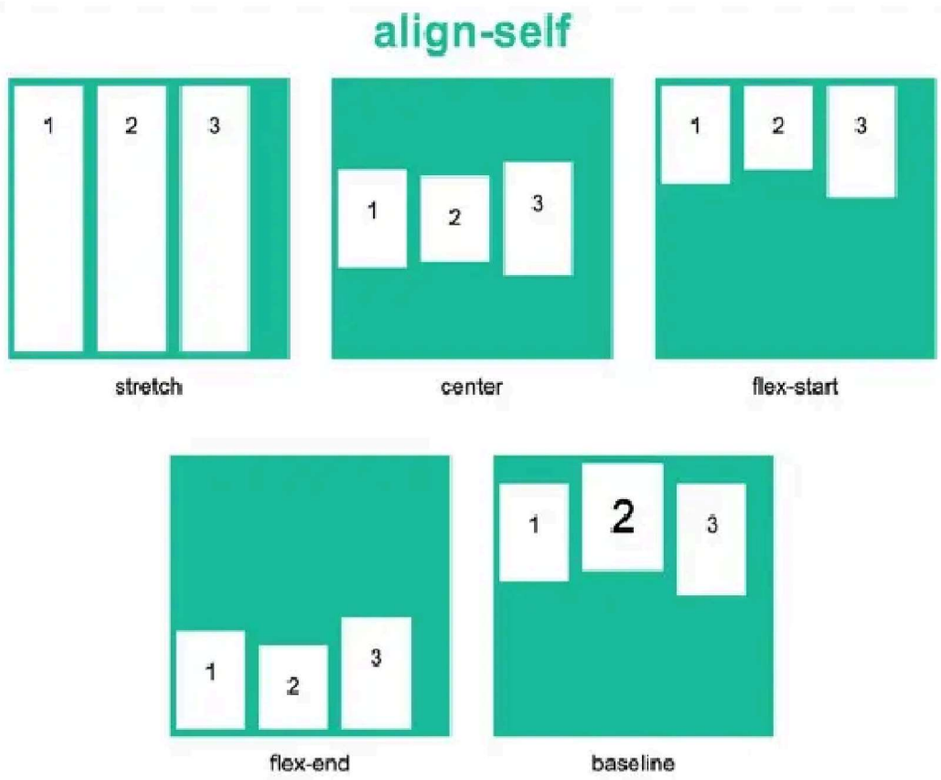
flex-basis tương tự như width và height, nó sẽ là kích thước theo chiều **main-axis**

Thuộc tính **flex** là một thuộc tính viết tắt cho thuộc tính **flex-grow**, **flex-shrink** và **flex-basis**

```
<div class="flex-container">
   <div>1</div>
   <div>2</div>
   <div style="flex: 0 0 200px">3</div>
   <div>4</div>
</div>
```

Thuộc tính **align-self** chỉ định căn item theo chiều cross axis. Các giá trị có thể sử dụng bao gồm:

- auto (mặc định): Sử dụng align-items của container
- flex-start
- flex-end
- center
- baseline



Hình 4-28 Thuộc tính align-self

Ví dụ căn giữa phần tử thứ 3:

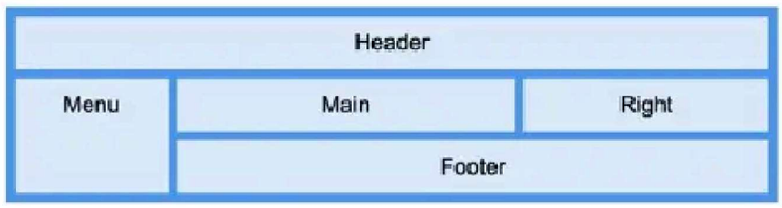
```
<div class="flex-container">
   <div>1</div>
   <div>2</div>
```

```
    <div style="align-self: center">3</div>
    <div>4</div>
</div>
```

4.12 LƯỚI (GRID)

4.12.1 Khái niệm lưới (Grid)

- Lưới cung cấp một hệ thống bố cục dựa trên lưới, với các hàng và cột, giúp thiết kế trang Web dễ dàng hơn.
- Bố cục lưới bao gồm một phần tử cha (container), với một hoặc nhiều phần tử con (item).



Ví dụ:

```
<!DOCTYPE html>
<html>
<head>
    <style>
        .grid-container {
            display: grid;
            grid-template-columns: auto auto auto ;
        }

        .grid-item {
            background-color: antiquewhite;
            border: 1px solid rgba(0, 0, 0, 0.8);
            padding: 10px;
            margin: 5px;
            font-size: 30px;
            text-align: center;
        }
    </style>
</head>
<body>

<div class="grid-container">
    <div class="grid-item">1</div>
    <div class="grid-item">2</div>
    <div class="grid-item">3</div>
    <div class="grid-item">4</div>
    <div class="grid-item">5</div>
    <div class="grid-item">6</div>
</div>

</body>
</html>
```

4.12.2 Grid Container

Tạo một grid **container** có thể được tạo bằng cách sử dụng thuộc tính **display:**

- grid
- inline-grid

Một số thuộc tính khác thiết lập cho container bao gồm:

- grid-template-columns
- grid-template-rows
- grid-template
- grid-template-areas
- justify-content
- align-content
- grid-column-gap
- grid-row-gap
- grid-gap
- grid-column-start
- grid-column-end
- grid-row-start
- grid-row-end

Thuộc tính **grid-template-columns** xác định số cột và độ rộng của bố cục lưới. Ví dụ 2 cột với kích thước tự động:

```
.grid-container {
   display: grid;
   grid-template-columns: auto auto;
}
```

Cột đầu có 150px, cột thứ 2 là 200px

```
.grid-container {
   display: grid;
   grid-template-columns: 150px 200px;
}
```

Thuộc tính **grid-template-rows** xác định kích thước của mỗi hàng. Ví dụ:

```
.grid-container {
   display: grid;
   grid-template-columns: auto auto auto auto;
   grid-template-rows: 100px 300px;
}
```

Thuộc tính **grid-template** xác định số cột và độ rộng cột trên một thuộc tính duy nhất. Đây là cách viết rút gọn của **grid-template-rows** và **grid-template-column.**

```
.grid-container {
   display: grid;
   grid-template: 150px 250px / auto auto auto;
}
```

Thuộc tính **grid-template-areas** xác định các vùng trong grid-layout, ví dụ:

```
.item1 {
   grid-area: myArea;
}
.grid-container {
   display: grid;
   grid-template-areas:
       'myArea myArea .'
       'myArea myArea .'
}
```

Thuộc tính **justify-content** thiết lập căn các item trong container. Các giá trị có thể được thiết lập bao gồm:

- unset (mặc định)
- space-evenly
- space-around
- space-between
- center
- start
- end

Ví dụ:

```
.grid-container {
   display: grid;
   grid-template-columns: auto auto auto;
   justify-content: space-evenly;
}
```

Thuộc tính **align-content** thiết lập căn theo chiều dọc. Các giá trị có thể thiết lập cho thuộc tính này:

- center
- space-evenly
- space-around
- space-between
- start
- end

Ví dụ:

```
.grid-container {
   display: grid;
```

```
    background: beige;
    grid-template-columns: auto auto auto;
    height: 300px;
    align-content: end;
}
```

Thuộc tính **grid-column-gap** dùng để thiết lập khoảng cách giữa các cột. Ví dụ:

```
.grid-container {
    display: grid;
    background: beige;
    grid-template-columns: auto auto auto;
    height: 300px;
    grid-column-gap: 50px;
}
```

Thuộc tính **grid-row-gap** dùng để thiết lập khoảng cách giữa các hàng. Ví dụ:

```
.grid-container {
    display: grid;
    background: beige;
    grid-template-columns: auto auto auto;
    height: 300px;
    grid-row-gap: 50px;
}
```

Thuộc tính **grid-gap** dùng để viết tắt cho 2 thuộc tính **grid-column-gap** và **grid-row-gap**. Ví dụ:

```
.grid-container {
    display: grid;
    background: beige;
    grid-template-columns: auto auto auto;
    height: 300px;
    grid-gap: 50px 100px;
}
```

4.12.3 Các phần tử grid Item

Trong container có các phần tử con (item), các phần tử đó có thể thiết lập một số các thuộc tính như sau:

- grid-column-start
- grid-column-end
- grid-row-start
- grid-row-end
- grid-column
- grid-row
- grid-area

Thuộc tính **grid-column-start** và **grid-column-end** và dùng để thiết lập vị trí bắt đầu và kết thúc cột. Ví dụ:

```
<div class="grid-container">
   <div class="grid-item" style="grid-column-start: 1; grid-column-end:
4">1</div>
   <div class="grid-item">2</div>
   <div class="grid-item">3</div>
   <div class="grid-item">4</div>
   <div class="grid-item">5</div>
   <div class="grid-item">6</div>
</div>
```

Thuộc tính **grid-column-start** và **grid-column-end** và dùng để thiết lập vị trí bắt đầu và kết thúc hàng. Ví dụ:

```
<div class="grid-container">
   <div class="grid-item" style="grid-row-start: 1; grid-row-end: 4">1</div>
   <div class="grid-item">2</div>
   <div class="grid-item">3</div>
   <div class="grid-item">4</div>
   <div class="grid-item">5</div>
   <div class="grid-item">6</div>
</div>
```

Thuộc tính **grid-column** xác định **số lượng cột** để đặt một item. Ví dụ sử dụng 2 con số để xác định cột bắt đầu và kết thúc ở đâu:

```
<div class="grid-container">
   <div class="grid-item" style="grid-column: 1 / 3">1</div>
   <div class="grid-item">2</div>
   <div class="grid-item" >3</div>
   <div class="grid-item" style="grid-column: 2 / 4">4</div>
   <div class="grid-item" style="grid-column: 1 / 3">5</div>
   <div class="grid-item" >6</div>
</div>
```

Ví dụ khác, sử dụng số và span độ dài để biết cột bắt đầu ở đâu và kéo dài mấy cột:

```
<div class="grid-container">
   <div class="grid-item" style="grid-column: 1 / span 3">1</div>
   <div class="grid-item">2</div>
   <div class="grid-item" >3</div>
   <div class="grid-item">4</div>
   <div class="grid-item" style="grid-column: 1 / span 2">5</div>
   <div class="grid-item" >6</div>
</div>
```

Thuộc tính **grid-row** xác định hàng nào sẽ đặt một mục. Ví dụ:

```
<div class="grid-container">
   <div class="grid-item" style="grid-column: 1 / span 3">1</div>
   <div class="grid-item" style="grid-row: 2 / span 2">2</div>
   <div class="grid-item" >3</div>
   <div class="grid-item">4</div>
   <div class="grid-item">5</div>
   <div class="grid-item" >6</div>
</div>
```

Thuộc tính **grid-area** có thể được sử dụng như một thuộc tính viết tắt cho các thuộc tính **grid-row-start**, **grid-column-start**, **grid-row-end** và **grid-column-end**. Ví dụ

```
<div class="grid-container">
   <div class="grid-item">1</div>
   <div class="grid-item">2</div>
   <div class="grid-item">3</div>
   <div class="grid-item">4</div>
   <div class="grid-item">5</div>
   <div class="grid-item">6</div>
   <div class="grid-item" style="grid-area: 1 / 2 / 4 / 3">7</div>
</div>
```

e. Đặt tên Items

Ngoài ra còn sử dụng kỹ thuật đặt tên để tạo bố cục như sau:

```
<!DOCTYPE html>
<html>
<head>
   <style>
       .item1 { grid-area: header; }
       .item2 { grid-area: menu; }
       .item3 { grid-area: main; }
       .item4 { grid-area: right; }
       .item5 { grid-area: footer; }

       .grid-container {
           display: grid;
           background: beige;
           grid-template-areas:
               'header header header header header header'
               'menu main main main right right'
               'menu footer footer footer footer footer';
       }

       .grid-container > div {
           background-color: antiquewhite;
           border: 1px solid rgba(0, 0, 0, 0.8);
           padding: 10px;
           margin: 5px;
           font-size: 30px;
           text-align: center;
       }
   </style>
</head>
<body>

<div class="grid-container">
   <div class="item1">Header</div>
   <div class="item2">Menu</div>
   <div class="item3">Main</div>
   <div class="item4">Right</div>
   <div class="item5">Footer</div>
</div>

</body>
```

```
</html>
```

4.13 BỐ CỤC WEBSITE

4.13.1 Bố cục Website

Một trang Web thường được chia thành các phần **header**, **menu**, **content** và **footer**. Trong đó phần **content** có thể chứa thêm các **aside**.

Có rất nhiều bộ cục thiết kế khác nhau để lựa chọn, tuy nhiên, bố cục sử dụng 3 cột như trên là một trong những bố cục phổ biến nhất.

Việc sử dụng thiết kế bố cục Website có thể sử dụng **flexbox** bằng cách tạo container:

```
<div class="flex-container">
  …
</div>
```

```
.flex-container {
   display: flex;
   flex-wrap: wrap;
}
```

4.13.2 Header

Phần đầu của Website thường nằm ở phía trên cùng của trang Web (hoặc ngay phía dưới menu điều hướng). Nó thường chứa một logo hoặc tên Website:

```
<header>Header</header>
```

```
header{
   flex-basis: 100%;
   background-color: gainsboro;
   padding: 20px;
   margin: 5px;
}
```

4.13.3 Thanh điều hướng

Một thanh điều hướng chứa một danh sách các liên kết để giúp người dùng điều hướng tới các mục khác nhau của Website.

```
<nav><a href="#">Link</a>
   <a href="#">Link</a>
   <a href="#">Link</a>
</nav>
```

```
nav{
   flex-basis: 100%;
   background-color: gainsboro;
   margin: 5px;
   display: flex;
}
nav a{
   padding: 20px;
   overflow: auto;
   text-decoration: none;
   color: black;

}
nav a:hover {
   background-color: #333;
   color: white;
}
```

4.13.4 Content

Cách bố trí trong phần này thường phụ thuộc vào mục đích của người dùng. Các bố trí phổ biến nhất là sử dụng các cột (thường từ 1 đến 3 cột):

- 1 cột (thường dùng cho trình duyệt di động).
- 2 cột (thường sử dụng cho máy tính bảng và máy tính xách tay).
- 3 cột (chỉ được sử dụng cho máy tính để bàn).

Các nhà thiết kế Web thường sử dụng bố cục 3 cột và thay đổi nó thành bố cục 1 hoặc 2 cột trên trên màn hình nhỏ hơn bằng cách sử dụng **@media**.

Ví dụ một thiết kế 3 cột:

```
<aside class="left">
   Aside Left
</aside>
<article>
   Main Content
</article>
<aside class="right">
   Aside Right
</aside>
```

```
aside.left {
   background: #F1F1F1;
   flex: 1 0;
   padding: 20px;
   margin: 5px;
}
article {
   text-align: left;
   background: #F1F1F1;
   flex: 3 0;
   padding: 20px 20px 100px;
   margin: 5px;
}
aside.right {
   background: #F1F1F1;
   flex: 1 0;
   padding: 20px;
   margin: 5px;
}
```

4.13.5 Footer

Footer được đặt dưới cuối trang Web, thường chứa thông tin như bản quyền và thông tin liên hệ:

```
<footer>Footer</footer>
```

```
footer{
   background-color: gainsboro;
   flex-basis: 100%;
   padding: 20px;
   margin: 5px;
}
```

4.14 THIẾT KẾ WEB ĐÁP ỨNG

4.14.1 Thiết kế Web đáp ứng là gì?

Thiết kế Web đáp ứng làm cho trang Web hiển thị hợp lý để làm tăng trải nghiệm cho tất cả người dùng sử dụng trên các thiết bị khác nhau:

- Máy tính để bàn
- Máy tính bảng
- Điện thoại

Các trang Web có kích thước cố định thường là quá lớn để phù hợp với người dùng sử dụng thiết bị di động. Để khắc phục điều này, các trình duyệt trên các thiết bị này tự động thu nhỏ toàn bộ trang Web để vừa với màn hình. Khi chiều ngang của thiết bị quá nhỏ, người dùng phải vuốt ngang để xem hết nội dung của trang Web hoặc xem trang Web với nội dung quá nhỏ và

cần phải zoom để đọc được nội dung. Điều này khiến người dùng cảm thấy không thoải mái khi lướt Web. Chính vì vậy cần Thiết kế Web đáp ứng nhằm tăng trải nghiệm của người dùng.

Thiết kế Web đáp ứng sử dụng CSS và HTML, đôi khi kết hợp thêm JavaScript để thay đổi kích thước, ẩn, thu nhỏ, phóng to hoặc di chuyển nội dung để làm cho nó trông đẹp hơn trên bất kỳ màn hình nào.

4.14.2 Viewport

- **Khung nhìn (Viewport)** là khu vực mà người dùng có thể nhìn thấy nội dung của trang Web.
- **Viewport** của từng thiết bị sẽ khác nhau, viewport của màn hình trên thiết bị di động thường sẽ nhỏ hơn khác nhiều so với máy tính bảng, máy bàn.
- Để thiết kế Website cho các thiết bị di động cần thiết lập **Viewport**.
- **Viewport** là thông tin được khai báo bởi phần tử **meta** trong phần tử **head.**

Ví dụ một Website không có viewport và cùng một trang Web có khai báo viewport:

Khai báo viewport:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0>
```

- **width**: Định dạng chiều rộng của viewport. thường là **Device-width**
- **initial-scale**: Định dạng mức phóng to trình duyệt lúc ban đầu

4.14.3 Media Queries

Media queries có thể được sử dụng để kiểm tra kích thước của viewport, từ đó áp dụng các CSS phù hợp. Cú pháp:

```
@media not|only mediatype and (expressions) {
  CSS-Code;
}
```

Ví dụ áp dụng cho CSS cụ thể:

```
@media only screen and (max-width: 1000px) and (min-width: 600px)  {
   body {
       background-color: lightblue;
   }
}
```

Ví dụ áp dụng cho file CSS cụ thể:

```
<link rel="stylesheet" media="mediatype and|not|only (expressions)"
href="print.css">
```

mediatype được sử dụng để giới hạn CSS sẽ được áp dụng cho màn hình, có thể thiết lập các giá trị khác như:

Giá trị	Mô tả
all	Sử dụng cho tất cả các loại thiết bị phương tiện truyền thông
print	Sử dụng cho máy in
screen	Sử dụng cho màn hình máy tính, máy tính bảng, điện thoại thông minh
speech	Được sử dụng cho các trình đọc màn hình “read”

expression gồm các thuộc tính sau:

- aspect-ratio: Tỉ lệ giữa chiều rộng và chiều cao của viewport.
- min-aspect-ratio: Tỉ lệ tối thiểu giữa chiều rộng và chiều cao của viewport.
- max-aspect-ratio: Tỉ lệ tối đa giữa chiều rộng và chiều cao của viewport.
- color: Số bits cho mỗi màu sắc của thiết bị.
- color-index: Số lượng màu sắc mà device có thể hiển thị.
- device-aspect-ratio: Tỉ lệ giữa chiều rộng và chiều cao của thiết bị.
- max-device-aspect-ratio: Tỉ lệ tối đa giữa chiều rộng và chiều cao của thiết bị.
- min-device-aspect-ratio: Tỉ lệ tối thiểu giữa chiều rộng và chiều cao của thiết bị.
- device-height: Chiều cao của device.
- device-width: Chiều rộng của device.
- height: Chiều cao của viewport.

- width: Chiều rộng của viewport.
- max-width: Chiều rộng tối đa của viewport (*).
- min-width: Chiều rộng tối thiểu của viewport (*).
- max-height: Chiều cao tối đa của viewport.
- min-height: Chiều cao tối thiểu của viewport.
- min-device-width: Chiều rộng tối thiểu của device.
- max-device-width: Chiều rộng tối đa của device.
- min-device-height: Chiều cao tối thiểu của device.
- max-device-height: Chiều cao tối đa của device.
- orientation: Định hướng của khung nhìn (xoay hoặc không xoay thiết bị).
- resolution: Độ phân giải của thiết bị đầu ra (sử dụng dpi hoặc dpcm).

Ví dụ: Nếu chiều rộng của trình duyệt nhỏ hơn hoặc bằng 600px thì bỏ margin xung quanh:

```
@media only screen and (max-width: 600px) {
   body {
      margin: 0;
   }
}
```

Có thể điều chỉnh số cột dựa vào kích thước ví dụ:

```
@media (max-width: 800px) {
   aside.left {
       order: 2;
   }
   article {
       order: 1;
       flex-basis: 100%;
   }
   aside.right {
       order: 3;
   }
   footer {
       order: 4;
   }
}
@media (max-width: 600px) {
   nav{
       flex-direction: column;
   }
   aside.left {
       order: 2;
       flex-basis: 100%;
   }
   article {
       order: 1;
       flex-basis: 100%;
   }
   aside.right {
       order: 3;
```

```
        flex-basis: 100%;
    }
    footer {
        order: 4;
    }
}
```

4.14.4 Thiết kế "Mobile First"

Mobile First là cách thiết kế cho di động trước, rồi áp dụng các Media Query cho các thiết bị có kích thước lớn. **Mobile first** là cách thiết kế phổ biến vì **Mobile first** khiến cho tốc độ hiển thị trên di động sẽ nhanh hơn.

```
.flex-container {
    display: flex;
    flex-wrap: wrap;
}
header{
    background-color: gainsboro;
    margin: 5px;
    flex-basis: 100%;
    padding: 20px;
}
nav{
    flex-basis: 100%;
    background-color: gainsboro;
    margin: 5px;
    margin: 5px;
    display: flex;
}
nav a{
    padding: 20px;
    overflow: auto;
    text-decoration: none;
    color: black;

}
nav a:hover {
    background-color: #333;
    color: white;
}
aside.left {
    background: gainsboro;
    order: 2;
    flex-basis: 100%;
    padding: 20px;
    margin: 5px;
}
article {
    background: gainsboro;
    order: 1;
    flex-basis: 100%;
    padding: 20px 20px 100px;
    margin: 5px;
```

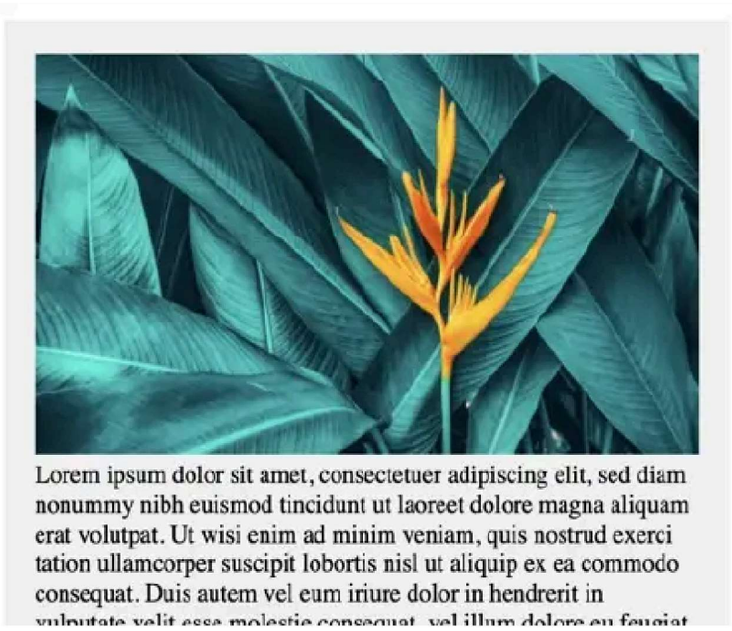
```
    text-align: left;
}
aside.right {
    background: gainsboro;
    order: 3;
    flex-basis: 100%;
    padding: 20px;
    margin: 5px;
}
footer {
    background: gainsboro;
    flex-basis: 100%;
    padding: 20px;
    margin: 5px;
    order: 4;
}
@media (min-width: 800px) {
    aside.left {
        order: 2;
        flex-basis: 0;
    }
    article {
        order: 1;
        flex-basis: 0;
    }
    aside.right {
        order: 3;
        flex-basis: 0;
    }
    footer {
        order: 4;
    }
}
@media (min-width: 800px) {
    aside.left {
        order: 1;
        flex-grow: 1;
    }
    article {
        order: 2;
        flex-grow: 3;
    }
    aside.right {
        order: 3;
        flex-grow: 1;
    }
}
```

4.14.5 Responsive Images & Videos

Khi hiển thị ảnh nội dung trên các thiết bị di động, nên sử dụng ảnh với kích thước bằng kích thước màn hình để ảnh hiển thị được rõ:

```
article img {
    width: 100%;
```

```
}
```



Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in

Khi hiển thị trên máy tính, ảnh có thể chỉ cần chiếm một phần màn hình, ví dụ:

```
@media (min-width: 800px) {
   article img {
       width: 240px;
       margin-right: 20px;
       margin-bottom: 10px;
       float: left;
   }
}
```

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum.

Trường hợp muốn sử dụng nhiều hình ảnh khác nhau để cho các màn hình có kích thước khác nhau có thể sử dụng đến phần tử HTML <picture>. Ví dụ:

```
<picture>
  <source media="(min-width:650px)" srcset="img_pink_flowers.jpg">
  <source media="(min-width:465px)" srcset="img_white_flower.jpg">
```

```
  <img src="img_orange_flowers.jpg" alt="Flowers" style="width:auto;">
</picture>
```

Phần tử <picture> trong ví dụ trên kết hợp với việc khai báo phần tử <source> và phần tử <img>. Tạo ra một khung tranh đáp ứng, tùy vào kích thước màn hình mà sử dụng các bức ảnh khác nhau phù hợp với kích thước đó.

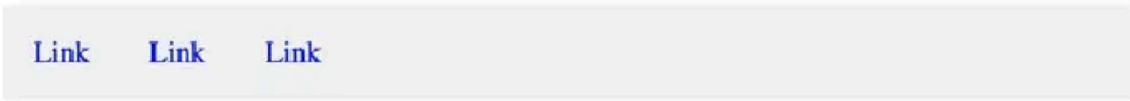
4.14.6 Responsive Menu

Menu đối với điện thoại thường to để dễ thực hiện các thao tác bấm.

```
nav{
   flex-basis: 100%;
   background-color: gainsboro;
   margin: 5px;
   margin: 5px;
   display: flex;
   flex-direction: column;
   text-align: center;
}
```

Máy tính có màn hình lớn vì vậy menu thường hiển thị theo hàng.

```
@media (min-width: 800px) {
   nav{
       flex-direction: row;
   }
}
```



Có thể tạo Hamburger menu bằng cách sử dụng checkbox

```
<label for="hamburger" id="hamburger-label"> &#9776;</label>
<input type="checkbox" id="hamburger"/>
<nav><a href="#">Link</a>
   <a href="#">Link</a>
   <a href="#">Link</a>
</nav>
```

```
nav{
    flex-basis: 100%;
    background-color: gainsboro;
    margin: 5px;
    margin: 5px;
    display: flex;
    flex-direction: column;
    text-align: center;
    display: none; /* Bổ sung */
}
#hamburger-label { /* Bổ sung */
    width: 100%;
    font-style: normal;
    font-size: 1.2em;
    padding: 10px;
    margin: 5px;
    background-color: gainsboro;
    text-align: center;
}
input:checked ~ nav { /* Bổ sung */
    display: flex;
    flex-direction: column;
}
#hamburger { /* Bổ sung */
    display: none;
}
```

Đối với màn hình có kích cỡ lớn thì hiển thị menu như bình thường và ẩn hamburger đi, và hiển thị các liên kết trong menu:

```
@media (min-width: 800px) {
    nav{
        flex-direction: row;
        display: flex;
    }
    #hamburger-label {
        display: none;
    }
}
```

4.14.7 Ẩn các phần tử không cần thiết

Trong một số trường hợp có thể phải ẩn một số thành phần không cần thiết khi hiển thị trên các thiết bị có màn hình nhỏ.

```
@media screen and (max-width: 600px) {
  header.logo {
      display: none;
  }
}
```

4.14.8 Cỡ chữ

Có thể thay đổi kích thước font chữ để đáp ứng cho các màn hình có thiết bị khác nhau.

```
@media screen and (min-width: 600px) {
  header {
       font-size: 80px;
  }
}

@media screen and (max-width: 600px) {
  header {
       font-size: 30px;
  }
```

Ngoài ra cũng có thể sử dụng các đơn vị tương đối để xác định kích thước của font chữ.

THỰC HÀNH

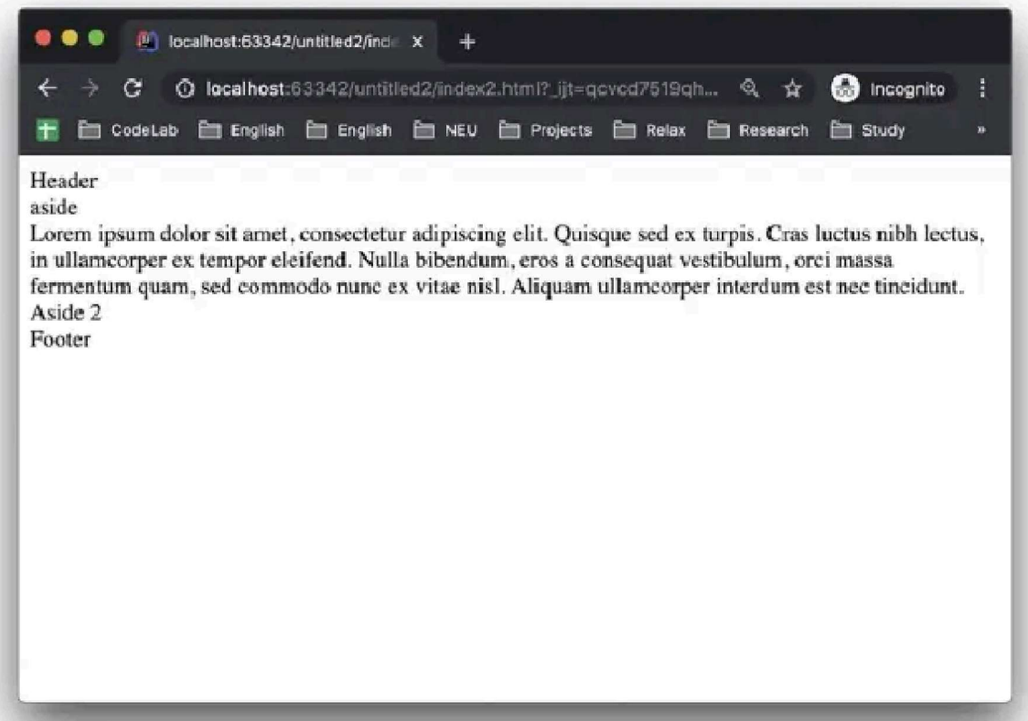
Bài 1. Thiết kế Website 1

Thiết kế Website có nội dung như sau:

Tạo layout ban đầu

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
<div class="flex-container">
   <header >Header</header>
   <aside class="aside1">aside </aside>
   <article>
       Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque sed ex
turpis. Cras
       luctus nibh lectus, in ullamcorper ex tempor eleifend. Nulla bibendum,
eros a consequat vestibulum, orci massa
       fermentum quam, sed commodo nunc ex vitae nisl. Aliquam ullamcorper
interdum est nec tincidunt.
   </article>
   <aside class="aside2">Aside 2</aside>
   <footer>Footer</footer>
</div>
</body>
</html>
```

Chạy thử và xem kết quả:



Sửa Thêm style màu sắc và flex-container

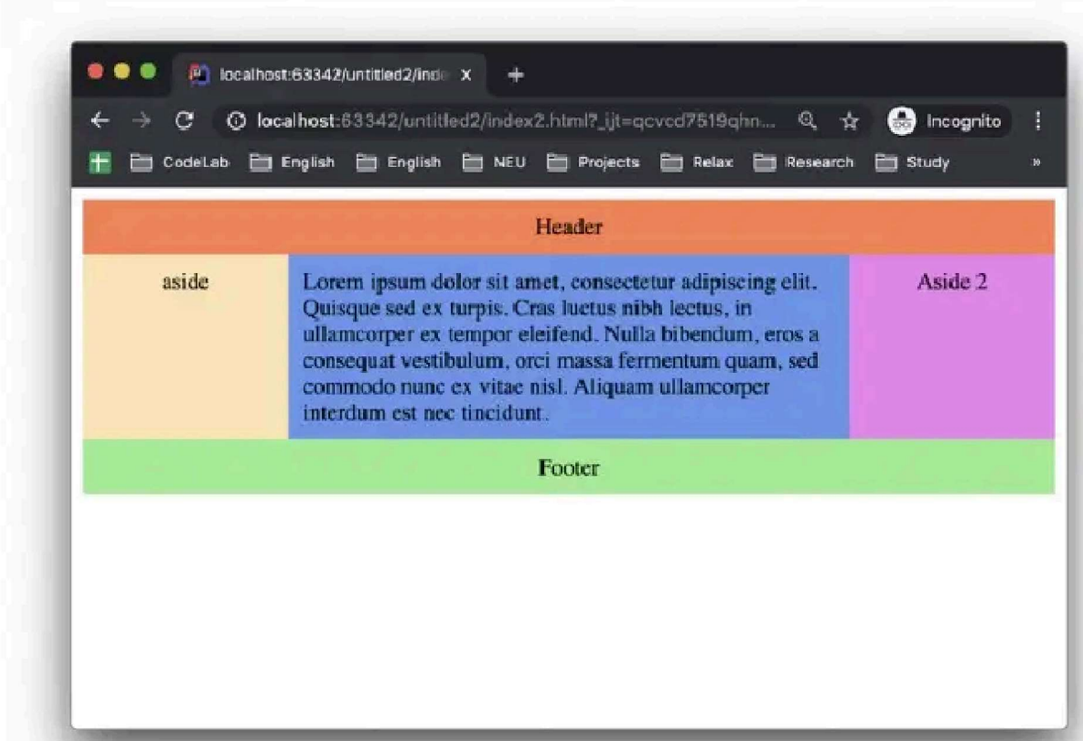
```
<style>
   .flex-container {
       display: flex;
       flex-flow: row wrap;
       text-align: center;
   }

   .flex-container > * {
       padding: 10px;
   }

   header {
       background: coral;
   }

   .aside1 {
       background: moccasin;
   }

   article {
       text-align: left;
       background: cornflowerblue;
   }
```

```
    .aside2 {
        background: violet;
    }

    footer {
        background: lightgreen;

    }
</style>
```

Thêm thuộc tính flex cho các khối

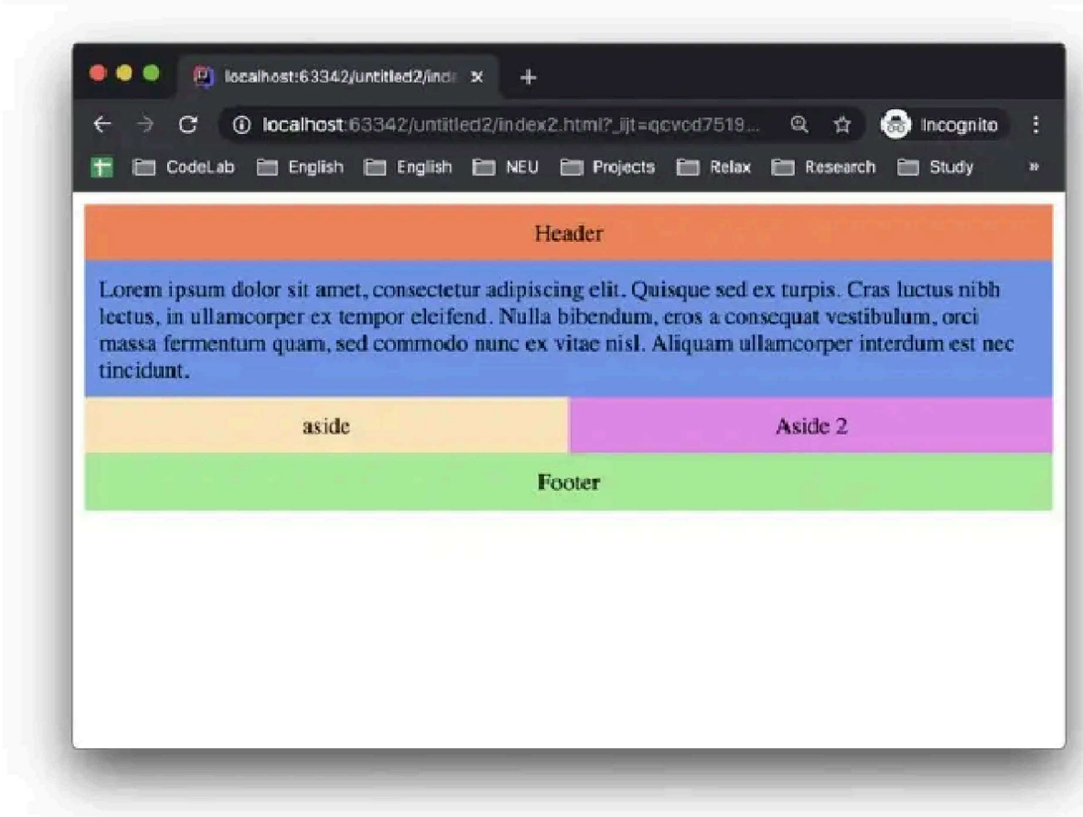
```
<style>
    .flex-container {
        display: flex;
        flex-flow: row wrap;
        text-align: center;
    }

    .flex-container > * {
        padding: 10px;
    }

    header {
        background: coral;
        flex: 1 100%;
    }

    .aside1 {
        background: moccasin;
        flex: 1 0px;
    }

    article {
        text-align: left;
        background: cornflowerblue;
        flex: 3 0px;
    }

    .aside2 {
        background: violet;
        flex: 1 0px;
    }

    footer {
        background: lightgreen;
        flex: 1 100%;
    }
</style>
```

Chạy thử và xem kết quả:



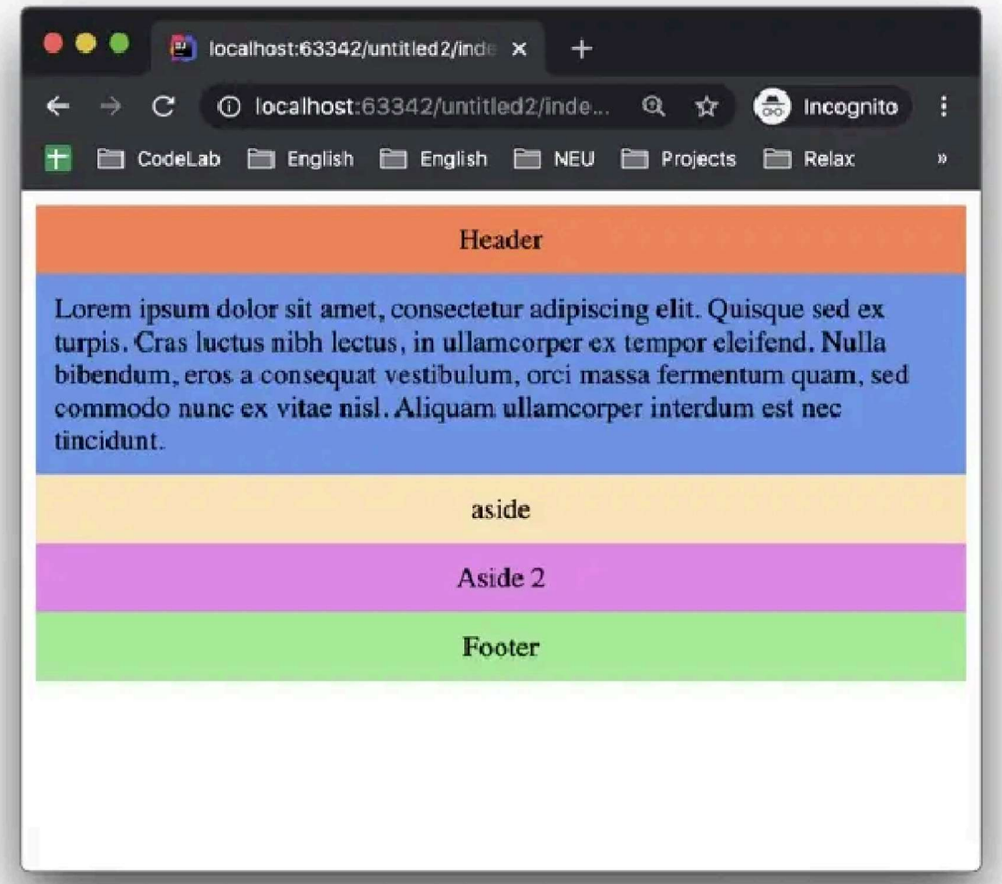
Responsive với kích thước 800px:

```
@media (max-width: 800px) {
   .aside1 {
       order: 2;
       flex: 1 0px;
   }
   article {
       order: 1;
       flex: 1 100%;
   }
   .aside2 {
       order: 3;
       flex: 1 0px;
   }
   footer {
       order: 4;
   }
}
```



Responsive với kích thước 600px:

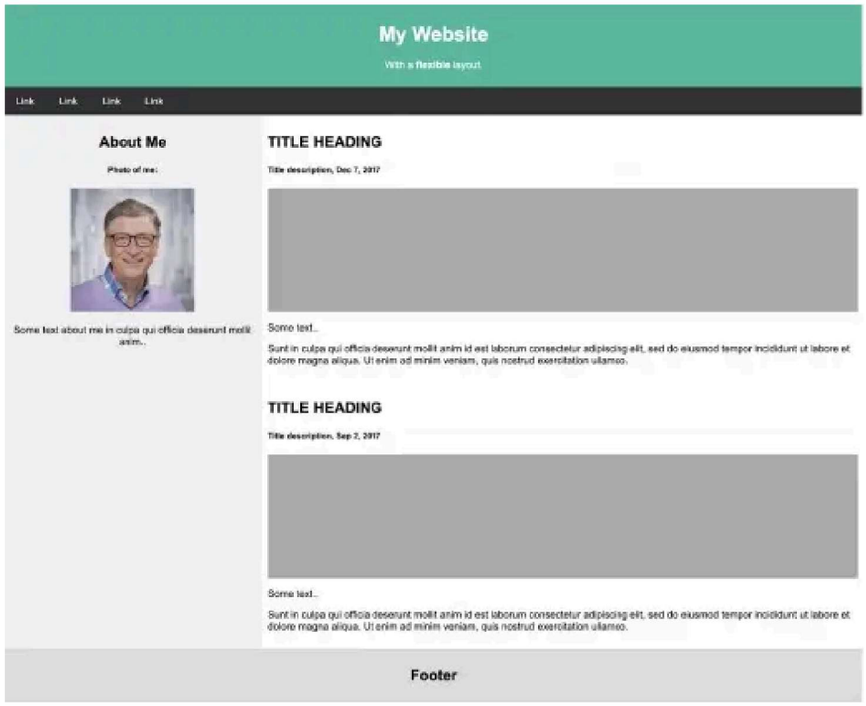
```
@media (max-width: 600px) {
   .aside1 {
       order: 2;
       flex: 1 100%;
   }
   article {
       order: 1;
       flex: 1 100%;
   }
   .aside2 {
       order: 3;
       flex: 1 100%;
   }
}
```



Toàn bộ code Website 1: **https://codepen.io/lampx83/pen/mdKJazb**

Bài 2. Thiết kế Website 2

Sửa Webiste 1 thành như sau:



Xóa aside 2 để website chỉ còn 2 cột, sửa lại Header với nội dung như sau:

```
<header>
   <h1>My Website</h1>
   <p>With a <b>flexible</b> layout.</p>
</header>

<footer>
   <h2>Footer</h2>
</footer>
```

Sửa lại Header, footer thay bằng style sau:

```
header {
   flex: 1 100%;
   background: #1abc9c;
   color: white;
   padding: 60px;
   text-align: center;
}
footer {
   flex: 100%;
   padding: 20px;
   text-align: center;
   background: #ddd;
}
```

Thay đổi font chữ và margin của website bằng style sau:

```
body {
   font-family: Arial;
   margin: 0;
```

```
}
```

Tạo NavBar sau Header với nội dung như sau:

```
<div class="navbar">
   <a href="#">Link</a>
   <a href="#">Link</a>
   <a href="#">Link</a>
   <a href="#">Link</a>
</div>
```

Định dạng cho Navbar:

```
.navbar {
   display: flex;
   background-color: #333;
   flex: 100%;
   padding: 0px;
}
.navbar a {
   color: white;
   padding: 14px 20px;
   text-decoration: none;
   text-align: center;
}
.navbar a:hover {
   background-color: #ddd;
   color: black;
}
```

Sửa định dạng Aside và article:

```
aside {
   flex: 30%;
   background-color: #f1f1f1;
   text-align: center;

}
article {
   flex: 70%;
   background-color: white;
}
```

Sửa lỗi nhảy xuống bằng cách cho thêm:

```
box-sizing: border-box;
```

Sửa lại Aside cho đẹp:

```
<aside class="aside1"><h2>About Me</h2>
   <h5>Photo of me:</h5>
   <img src="bill_gate.jpg" height="200px"/>
   <p>Some text about me in culpa qui officia deserunt mollit anim..</p>
</aside>
```

Sửa lại article:

```
<article>
   <h2>TITLE HEADING</h2>
   <h5>Title description, Dec 7, 2017</h5>
   <div class="fakeimg" style="height:200px;"></div>
   <p>Some text..</p>
   <p>Sunt in culpa qui officia deserunt mollit anim id est laborum consectetur
adipiscing elit, sed do eiusmod
       tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim
veniam, quis nostrud exercitation
       ullamco.</p>
   <br>
   <h2>TITLE HEADING</h2>
   <h5>Title description, Sep 2, 2017</h5>
   <div class="fakeimg" style="height:200px;"></div>
   <p>Some text..</p>
   <p>Sunt in culpa qui officia deserunt mollit anim id est laborum consectetur
adipiscing elit, sed do eiusmod
       tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim
veniam, quis nostrud exercitation
       ullamco.</p>
</article>
```

Thêm style :

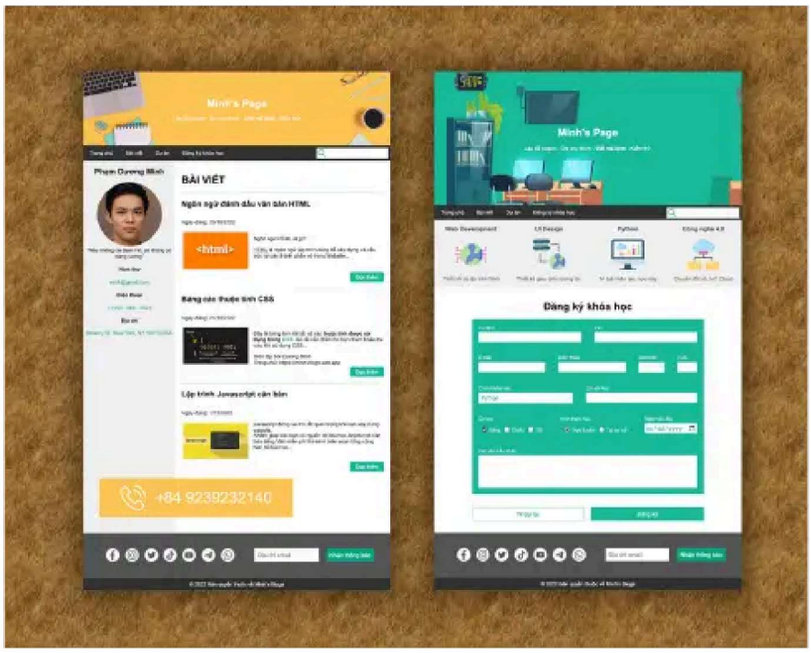
```
.fakeimg {
   background-color: #aaa;
   width: 100%;

}
```

Responsive cho màn hình nhỏ hơn 800:

```
@media screen and (max-width: 800px) {
   .flex-container, .navbar {
       flex-direction: column;
   }
}
```

Bài 3. Thiết kế Website MinhBlogs

Dựng Website có thiết kế như sau:



Sử dụng file ảnh tại:

https://www.dropbox.com/sh/vk7vy83yw71wm8i/AAD2XhLUaavxxlbtK6PoL3MMa

Thiết kế Website đáp ứng với kích thước < 800px và >800

Chuẩn bị 1 thư mục public bên trong có chứa

- file index.html
- Thư mục css bên trong có file style.css
- Thư mục js bên trong có file main.js
- Thư mục images bên trong chứa các ảnh cần thiết

Tạo file tài liệu index.html

```
<!DOCTYPE html>
<html lang="vi">
<head>
   <meta charset="UTF-8">
   <title>My Blogs</title>
   <link href="css/style.css" rel="stylesheet">
</head>
<body>

<div class="flex-container">
   <header class="feature1">
       <h1>Minh's Page</h1>
       <p>Lập kế hoạch - Ghi chú thích - <b>Viết mã lệnh</b> - Kiểm thử</p>
   </header>
   <nav>
   </nav>
   <section>
       <h2>Phạm Dương Minh</h2>
   </section>
   <main>
```

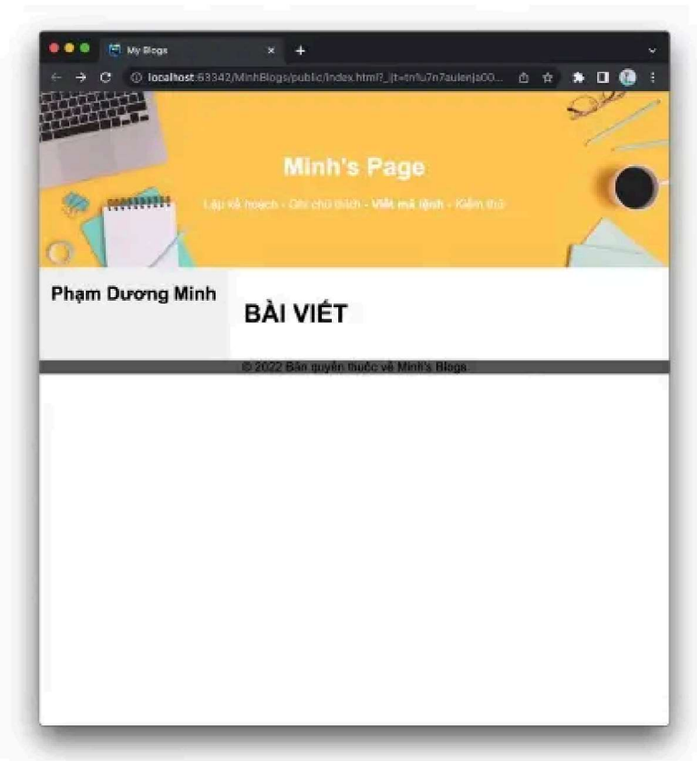
```
        <h1>BÀI VIẾT</h1>
    </main>
    <footer>
        &copy; 2022 Bản quyền thuộc về Minh's Blogs
    </footer>
</div>
</body>

</html>
```

Tạo file style.css

```
:root {
    --primary-color: #1abc9c;
    --primary-dark-color: #00a886
}
body {
    font-family: Arial;
    margin: 0
}
.flex-container {
    display: flex;
    flex-flow: row wrap
}
.flex-container > * {
    box-sizing: border-box
}

header {
    flex-grow: 1;
    flex-basis: 100%;
    color: #fff;
    text-align: center
}
header.feature1 {
    background: var(--primary-color) url(../images/feature1.jpg);
    background-size: cover;
    padding: 58px
}

nav {
    display: flex;
    background-color: #333;
    flex: 100%;
    padding: 0
}

section {
    flex: 30%;
    background-color: #f1f1f1;
    text-align: center
}

main {
    flex: 70%;
    background-color: #fff;
```

```
    display: flex;
    flex-direction: column;
    padding: 20px
}
footer {
    flex: 100%;
    text-align: center;
    background: #555
}
```

Kết quả



Phần Nav khai báo như sau trong file html:

```
<nav>
    <a href="index2.html">Trang chủ</a>
    <a href="#">Bài viết</a>
    <a href="#">Dự án</a>
    <a href="register.html">Đăng ký khóa học</a>
    <div class="search">
        <input type="search">
    </div>
</nav>
```

Bổ sung phần cấu hình nav:

```
nav div.search {
    flex-grow: 1;
```

```
   display: flex;
   justify-content: flex-end;
   padding: 5px
}

nav div.search input {
   outline: 0;
   font-size: 18px;
   background-image: url(../images/search.png);
   background-repeat: no-repeat;
   background-size: 24px 24px;
   background-position: 5px center;
   padding-left: 35px;
   color: var(--primary-dark-color)
}

nav div.search input:focus {
   width: 100%
}

nav a {
   color: #fff;
   padding: 14px 20px;
   text-decoration: none;
   text-align: center
}

nav a:hover {
   background-color: #ddd;
   color: #000
}
```

Chỉnh sửa Section:

```
<section>
   <h2>Phạm Dương Minh</h2>
   <img id="avatar" src="images/me.jpg" height="200" alt="Avatar">

   <q style="display: block">Nếu không có đam mê, sẽ không có năng lượng</q>
   <div class="contact">
       <h4 class="contact-title">Hòm thư</h4>
       <a href="mailto:(webmail@gmail.com)">minh@gmail.com</a>
       <h4 class="contact-title">Điện thoại</h4>
       <a href="tel:+1255-568-6523">+1255 - 568 - 6523</a>
       <h4 class="contact-title">Địa chỉ</h4>
       <a href="https://www.google.com/maps" target="blank">Bowery St, New York,
NY 10013,USA</a>
   </div>
</section>
```

Bổ sung phần định dạng section:

```
#avatar {
   border-radius: 50%
}
```

```
a {
   text-decoration: none
}

a {
   color: var(--primary-dark-color)
}

a:hover {
   text-decoration: underline
}
```

Chỉnh sửa phần Main, Sửa nội dung file index.html, mục main:

```
<main>
   <h1>BÀI VIẾT</h1>
   <article>
       <h2>Ngôn ngữ đánh dấu văn bản HTML</h2>
       <h3>Ngày đăng:
           <time datetime="2022-10-25">25/10/2022</time>
       </h3>
       <div class="content">
           <a href="article.html"><img src="images/html.jpg" alt="HTML"></a>
           <div>
               <p><em>Ngôn ngữ HTML là gì?</em></p>
               <p><abbr title="Hypertext Markup Language">HTML</abbr> là ngôn
ngữ lập trình dùng để xây dựng và cấu trúc lại các thành phần có trong
Website...</p>
           </div>
       </div>
       <a class="readmore">Đọc thêm</a>
   </article>
   <hr>
   <article>
       <h2>Bảng các thuộc tính CSS</h2>
       <h3>Ngày đăng:
           <time datetime="2022-10-21">21/10/2022</time>
       </h3>
       <div class="content">
           <a href="article.html"><img src="images/css.png" alt="CSS"></a>
           <div>
               <p>Đây là bảng tóm tắt tất cả các <span style="font-weight:
bold">thuộc tính được sử dụng trong <em style="color: #1abc9c">CSS</em></span>.
Sẽ rất cần thiết cho bạn tham
                   khảo,tra cứu khi sử dụng CSS...</p>
               <address>
                   Biên tập bởi Dương Minh<br> Trang chủ: https://minh-
blogs.web.app
               </address>
           </div>
       </div>
       <a class="readmore" href="article.html">Đọc thêm</a>
   </article>
   <hr>
   <article>
       <h2>Lập trình Javascript căn bản</h2>
```

```
        <h3>Ngày đăng:
            <time datetime="2022-12-01">1/12/2022</time>
        </h3>
        <div class="content">
            <a href="article.html"><img src="images/js.png" alt="Javascript"></a>
            <div>
                Javascript đóng vai trò rất quan trọng khi bạn xây dựng website.
<br>Nhằm giúp các bạn có nguồn
                tài liệu học Javascript căn bản tiếng Việt miễn phí thì mình biên
soạn tổng cộng hơn 50 bài
                học...
            </div>
        </div>
        <a class="readmore" href="article.html" title="Đọc thêm về
JavaScript">Đọc thêm</a>
    </article>
</main>
```

Bổ sung phần định dạng **main** trong file style.css

```
article {
    display: flex;
    flex-direction: column;
    border-top: 1px solid rgba(26, 188, 156, .5)
}

article .content {
    display: flex
}

article .content img {
    border: 1px solid #ddd;
    border-radius: 4px;
    padding: 5px;
    width: 200px
}

article .content img:hover {
    opacity: .75
}

article h3 {
    font-weight: 100;
    font-size: 1em
}

article .content div {
    padding-left: 10px
}

.readmore {
    align-self: flex-end;
    background-color: #1abc9c !important;
    border-radius: 5px;
    color: #fff;
    padding: 6px 18px;
```

```
    text-decoration: none
}
```

Sửa lại footer như sau:

```
<footer>
    <div class="footer-main">
        <img src="images/social.svg" alt="Social">
        <form>
            <input type="email" name="email" placeholder="Địa chỉ email">
            <input type="submit" value="Nhận thông báo">
        </form>

    </div>
    <div class="legal">
        &copy; 2022 Bản quyền thuộc về Minh's Blogs
    </div>
</footer>
```

Bổ sung định dạng các phần tử trong footer:

```
.footer-main {
    display: flex;
    justify-content: space-around;
    padding: 30px;
    align-items: center
}

.footer-main img {
    max-width: 400px;
    padding: 20px
}

footer form {
    margin: 0
}

footer input[type=email] {
    border: 0;
    height: 40px;
    outline: 0;
    font-size: 18px;
    padding-left: 10px
}

footer input[type=submit] {
    background-color: var(--primary-dark-color);
    color: #fff;
    border: 0;
    height: 40px;
    font-size: 18px;
}

.legal {
    padding: 10px;
    background-color: #333;
```

```
    color: #fff
}
```

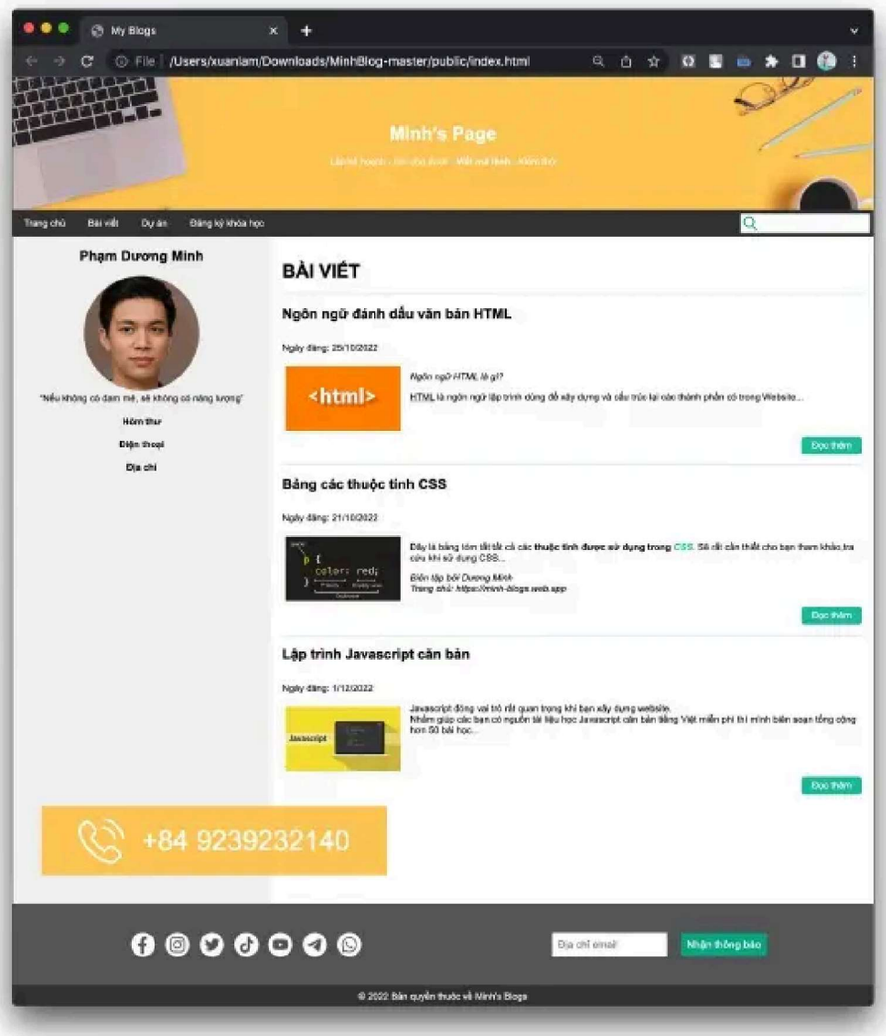
Bổ sung phần gọi điện, bằng cách thêm phần html sau trước footer và sau main:

```
<div class="space1">
    <div class="ad">
        <img src="images/call.svg" alt="Call"> +84 9239232140
    </div>
</div>
<div class="space2">
</div>
```

Bổ sung thêm css:

```
div.space1 {
    flex: 30%;
    background-color: #f1f1f1;
    height: 170px
}

div.space2 {
    flex: 70%;
    background-color: #fff;
    height: 170px
}

div.ad {
    display: flex;
    background-color: #ffc654;
    position: absolute;
    width: 600px;
    left: 50px;
    right: 50px;
    height: 120px;
    color: #fff;
    font-size: 3em;
    justify-content: center;
    align-items: center
}

div.ad img {
    fill: #fff;
    width: 80px;
    height: 80px;
    margin-right: 30px
}
```

Kết quả sau khi thiết kế như sau:



Tạo trang register.html

```
<!DOCTYPE html>
<html lang="vi">
<head>
   <meta charset="UTF-8">
   <meta name="description" content="Hướng dẫn thiết kế Website">
   <meta name="keywords" content="HTML, CSS, JavaScript ">
   <meta name="author" content="Dương Minh">
   <link rel="apple-touch-icon" sizes="180x180" href="images/favicon/apple-
touch-icon.png">
   <link rel="icon" type="image/png" sizes="32x32" href="images/favicon/favicon-
32x32.png">
   <link rel="icon" type="image/png" sizes="16x16" href="images/favicon/favicon-
16x16.png">
   <title>Đăng ký</title>
   <link href="css/style.css" rel="stylesheet">
</head>
```

```
<body>
<div class="flex-container">
   <header class="feature2">
       <h1>Minh's Page</h1>
       <p>Lập kế hoạch - Ghi chú thích - <b>Viết mã lệnh</b> - Kiểm thử</p>
   </header>
   <nav>
       <a href="index2.html">Trang chủ</a>
       <a href="#">Bài viết</a>
       <a href="#">Dự án</a>
       <a href="register.html">Đăng ký khóa học</a>
       <div class="search">
           <input type="search">
       </div>
   </nav>
   <div class="feature">
       4 khóa cơ bản
   </div>

   <main>
       Đăng ký khóa học
   </main>

   <footer>
       <div class="footer-main">
           <img src="images/social.svg" alt="Social">
           <form>
               <input type="email" name="email" placeholder="Địa chỉ email">
               <input type="submit" value="Nhận thông báo">
           </form>

       </div>
       <div class="legal">
           &copy; 2022 Bản quyền thuộc về Minh's Blogs
       </div>
   </footer>
```

Bổ sung css cho header của trang register:

```
header.feature2 {
   background: var(--primary-color) url(../images/feature2.png);
   background-size: cover;
   padding: 150px
}
```

Tạo phần 4 khóa học bằng cách Bổ sung mã html sau dưới nav:

```
<div class="feature">
   <div class="ad-course">
       <h3>Thiết kế Web</h3>
       <img src="images/a1.png" alt="Web Development">
       <p>Thiết kế và lập trình Web</p>
   </div>
   <div class="ad-course">
       <h3>UI Design</h3>
       <img src="images/a2.png"  alt="UI Design">
```

```
        <p>Thiết kế giao diện tương tác</p>
    </div>
    <div class="ad-course">
        <h3>Python</h3>
        <img src="images/a3.png"  alt="Python">
        <p>Trí tuệ nhân tạo, học máy</p>
    </div>
    <div class="ad-course">
        <h3>Công nghệ 4.0</h3>
        <img src="images/a4.png"  alt="Công nghệ 4.0">
        <p>Chuyển đổi số, IoT, Cloud</p>

    </div>
</div>
```

Bổ sung thêm các css:

```
div.feature {
    display: flex;
    background: #f1f1f1;
    flex-grow: 1;
    justify-content: space-around
}
div.feature div.ad-course {
    display: flex;
    flex-direction: column;
    color: #555;
    align-items: center
}

div.feature div.ad-course img {
    width: 100px;
    height: 100px
}
```

Bổ sung form sau trong main:

```
<form action="/submit.php" method="post" accept-charset="utf-8"
autocomplete="on">
    <input type="hidden" name="userID" value="23114306030131">
    <h1>Đăng ký khóa học</h1>
    <div class="form-block">
        <div class="form-subblock user-info">
            <div>
                <label for="last_name">Họ đệm</label>
                <input type="text" id="last_name" size="15">
            </div>
            <div>
                <label class="form-sub-label" for="last_name">Tên</label>
                <input type="text" id="first_name" size="15">
            </div>

        </div>
        <div class="form-subblock">
            <div>
                <label for="email">E-mail</label>
                <input type="email" id="email">
```

```
            </div>
            <div>
                <label for="phone">Điện thoại</label> <input type="tel"
id="phone" pattern="[0-9]{3}-[0-9]{2}-[0-9]{3}">
            </div>
            <div>
                <label class="form-label form-label-top" for="gender">Giới
tính</label>
                <select class="form-dropdown" id="gender">
                    <option value="Male">Nam</option>
                    <option value="Female">Nữ</option>
                    <option value="N/A">Khác</option>
                </select>
            </div>
            <div>
                <label for="age">Tuổi</label> <input type="number" id="age"
min="18" max="99">
            </div>
        </div>

        <div class="form-subblock">
            <div>
                <label for="course-select">Chọn khóa học:</label>
                <select id="course-select">
                    <optgroup label="Khóa học lập trình">
                        <option>Python</option>
                        <option>Java</option>
                        <option>JavaScript</option>
                    </optgroup>
                    <optgroup label="Cơ sở dữ liệu">
                        <option>MySQL</option>
                        <option>SQL Server</option>
                        <option>MongoDB</option>
                    </optgroup>
                </select>
            </div>
            <div>
                <label for="city">Cơ sở học:</label>
                <input list="cities" name="city" id="city">
                <datalist id="cities">
                    <option value="Hà Nội">
                    <option value="Thành phố Hồ Chí Minh">
                    <option value="Đà Nẵng">
                    <option value="Hải Phòng">
                    <option value="Cần Thơ">
                </datalist>
            </div>
        </div>
        <div class="form-subblock">
            <div>
                <label>Ca học</label>
                <div class="option">
                    <input type="checkbox" name="ca" value="male" id="sang"
checked><label for="sang">Sáng</label>
                    <input type="checkbox" name="ca" value="female"
id="chieu"><label for="chieu">Chiều</label>
```

```
                    <input type="checkbox" name="ca" value="other"
id="toi"><label for="toi">Tối</label>
                </div>
            </div>
            <div>
                <label>Hình thức học</label>
                <div class="option">
                    <input type="radio" name="ca" value="male" id="online"
checked><label for="online">Trực tuyến</label>
                    <input type="radio" name="ca" value="female"
id="offline"><label for="offline">Tại cơ sở</label>
                </div>
            </div>
            <div>
                <label>Ngày bắt đầu</label>
                <input type="date" name="start_date">
            </div>
        </div>
        <div class="form-subblock">
            <div>
                <label for="request">Các yêu cầu khác</label>
                <textarea id="request" name="request" style="display:
block"></textarea>
            </div>
        </div>
    </div>
    <div class="form-block-button" style="margin-top: 40px">
        <div>
            <input type="reset" value="Nhập lại">
        </div>
        <div>
            <input type="submit" value="Đăng ký">
        </div>
    </div>
</form>
```

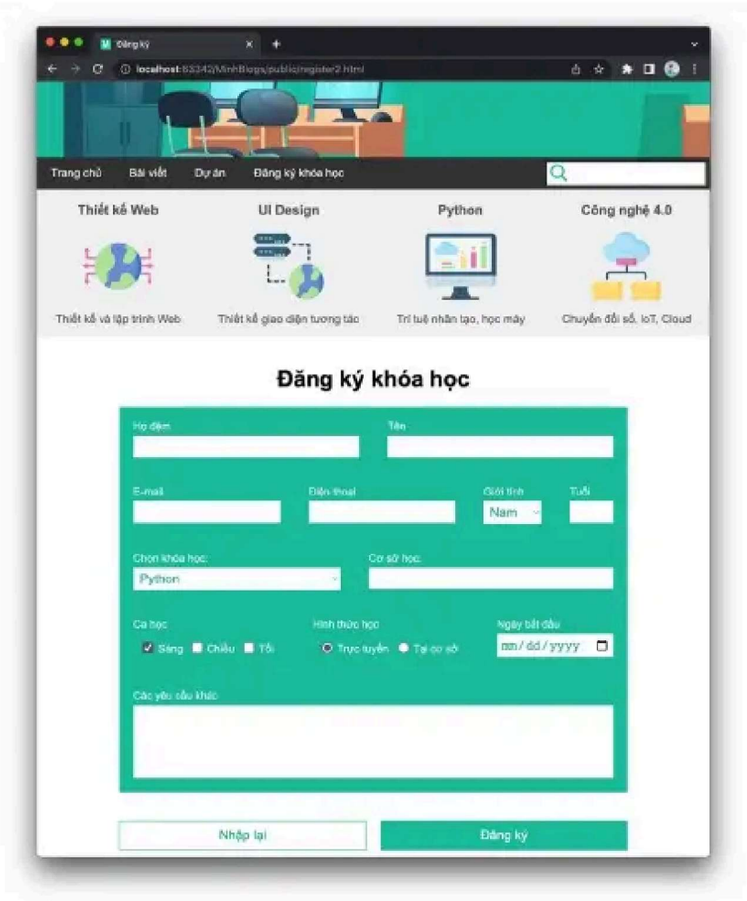
Bổ sung thêm các css:

```
main form {
    align-self: center
}

.form-block {
    background-color: var(--primary-color);
    display: flex;
    flex-direction: column
}

.form-block label {
    color: #fff;
    opacity: 85%;
    padding-bottom: 5px;
    font-size: 15px
}

.form-block input, select {
```

```
    padding: 5px;
    border: 0;
    color: var(--primary-dark-color);
    accent-color: var(--primary-dark-color);
    font-size: 18px
}

.form-block input, select:focus {
    outline-width: 0
}

.form-subblock {
    display: flex
}

.form-subblock > div {
    flex-grow: 1;
    display: flex;
    flex-direction: column;
    padding: 20px
}

.form-subblock .option {
    padding: 10px
}

.form-subblock .option label {
    padding: 5px
}

.form-subblock .option input {
    transform: scale(1.2);
    accent-color: #555
}

.form-subblock textarea {
    border: none;
    height: 100px;
    resize: none;
    font-size: 18px;
    font-family: Arial;
    color: var(--primary-dark-color)
}

.form-subblock textarea:focus {
    outline-width: 0
}

.form-block-button {
    display: flex;
    flex-direction: row
}

input[type=submit] {
    padding: 10px;
    font-size: 18px;
```

```
   border: 1px solid var(--primary-color);
   background-color: var(--primary-color);
   color: #fff;
   margin-left: 20px
}

input[type=reset] {
   padding: 10px;
   font-size: 18px;
   border: 1px solid var(--primary-color);
   background-color: #fff;
   color: var(--primary-dark-color)
}

.form-block-button > div {
   flex-grow: 1;
   display: flex;
   flex-direction: column
}

form h1 {
   text-align: center
}

div.space1 {
   flex: 30%;
   background-color: #f1f1f1;
   height: 170px
}

div.space2 {
   flex: 70%;
   background-color: #fff;
   height: 170px
}

div.ad {
   display: flex;
   background-color: #ffc654;
   position: absolute;
   width: 600px;
   left: 50px;
   right: 50px;
   height: 120px;
   color: #fff;
   font-size: 3em;
   justify-content: center;
   align-items: center
}

div.ad img {
   fill: #fff;
   width: 80px;
   height: 80px;
   margin-right: 30px
}
```

Kết quả khi chạy:



Thiết kế Responsive (*):

```
@media screen and (max-width: 800px) {
   .flex-container, .footer-main, nav {
       flex-direction: column
   }

   .space1, .space2 {
       display: none
   }

   section,div.feature {
       order: 1;
       padding: 20px
   }

   footer {
       order: 2
   }

   nav div.search input {
       flex-grow: 1;
       padding-top: 6px;
       padding-bottom: 6px;
```

```
    }

    div.feature {
        flex-direction: column;
    }

    header.feature2,header.feature1 {
        padding: 40px
    }
}
```

Hãy bổ sung thêm css để có thiết kế đáp ứng cho phần biểu mẫu

CÂU HỎI ÔN TẬP LÝ THUYẾT

1. Thuộc tính CSS nào được sử dụng để thiết lập độ rộng của một phần tử HTML?

A. width
B. height
C. margin
D. padding

2. Thuộc tính CSS nào được sử dụng để thêm khoảng cách giữa các phần tử HTML?

E. width
F. height
G. margin
H. padding

3. Đơn vị kích thước CSS nào là đơn vị tương đối?

I. px
J. em
K. mm
L. cm

4. Đơn vị kích thước CSS nào được dùng để định nghĩa kích thước theo tỷ lệ phần trăm của chiều cao của khung nhìn?

M. em

N. vmin

O. %

P. vh

5. Khi sử dụng flexbox, các phần tử con sẽ được sắp xếp như thế nào theo mặc định?

Q. theo chiều dọc

R. theo chiều ngang

S. các phần tử con không được sắp xếp theo thứ tự mặc định.

T. Không có đáp án nào chính xác.

6. Thuộc tính flex-wrap được sử dụng để làm gì?

U. Điều chỉnh chiều cao của phần tử.

V. Điều chỉnh chiều rộng của phần tử.

W. Điều chỉnh hướng sắp xếp của các phần tử con.

X. Điều chỉnh các phần tử con được xuống dòng, hoặc sang cột mới hay không

7. Thuộc tính flex được sử dụng để làm gì?

Y. Điều chỉnh chiều cao của phần tử.

Z. Điều chỉnh chiều rộng của phần tử.

AA. Điều chỉnh hướng sắp xếp của các phần tử con.

BB. Kết hợp các thuộc tính flex-grow, flex-shrink và flex-basis thành một thuộc tính duy nhất.

8. Grid là gì trong CSS?

CC. Một kiểu kích thước cho các phần tử HTML.

DD. Một phương pháp sắp xếp và bố trí các phần tử HTML.

EE.Một kiểu font chữ.

FF. Một kiểu màu nền cho các phần tử HTML.

9. Thuộc tính nào được sử dụng trong grid để chỉ định khoảng cách giữa các ô?

GG. grid-gap

HH. grid-spacing

II. grid-padding

JJ. grid-margin

10. Các thuộc tính nào được sử dụng trong grid để định nghĩa số cột và hàng?

KK. grid-template-columns và grid-template-rows

LL. grid-template-rows và grid-template-areas

MM. grid-template-columns và grid-template-areas

NN. grid-columns và grid-rows

11. Các thuộc tính nào được sử dụng trong grid để xác định kích thước của các cột và hàng?

OO. grid-column-start, grid-column-end, grid-row-start và grid-row-end

PP. grid-column-size, grid-row-size, grid-column-width và grid-row-height

QQ. grid-column-width, grid-row-height, grid-column-gap và grid-row-gap

RR. grid-column-span, grid-row-span, grid-column-gap và grid-row-gap

12. Có thể sử dụng grid để tạo layout responsive không?

SS. Có, nhưng chỉ có thể sử dụng các giá trị cố định cho kích thước cột và hàng.

TT. Không, grid không hỗ trợ layout responsive.

UU. Có, grid hỗ trợ layout responsive và các giá trị kích thước động.

VV. Có, nhưng chỉ có thể sử dụng phương thức grid-template-areas để tạo layout responsive.

BÀI TẬP TỰ THỰC HÀNH

Bài 1. Thiết kế biểu mẫu đáp ứng

Khi màn hình có độ rộng nhỏ hơn 600px, làm cho hai cột xếp chồng lên nhau thay vì cạnh nhau.

Bài 2. Thiết kế Website với 3 kích thước cho 3 thiết bị

Hãy thiết kế một Website có giao diện như sau:

Bài 3. Sử dụng Grid trong thiết kế bố cục

Hãy thiết kế lại 2 trang Website 1 và 2 trong phần bài tập thực hành trên bằng cách sử dụng Grid thay cho Flex.

TÀI LIỆU THAM KHẢO

[1] Pro CSS3 Layout Techniques (2016), Sam Hampton-Smith, Apress

[2] UX / UI Wireframe Design Sketchbook: Mobile, Tablet and Desktop templates for responsive designs with project planning, (2020), UX/UI Designer Books, Independently published

[3] Building Websites All-in-One For Dummies (2012), David Karlins, For Dummies

CHƯƠNG 5: LẬP TRÌNH JAVASCRIPT CĂN BẢN

Chương này sẽ giới thiệu các khái niệm căn bản về JavaScript, một trong những thành phần quan trọng trong phát triển ứng dụng Web. Sau khi hoàn thành chương này, người học có thể biết cách sử dụng các câu lệnh căn bản trong JavaScript, nhúng các câu lệnh JavaScript trong một trang Web bất kỳ.

5.1 TỔNG QUAN VỀ JAVASCRIPT

5.1.1 Khái niệm JavaScript

JavaScript là ngôn ngữ lập trình được sử dụng rộng rãi:

- Phía trang Web (phía người dùng).
- Phía máy chủ (với Nodejs).

JavaScript là một trong "3 ngôn ngữ" mà tất cả các nhà phát triển Web đều phải biết, trong đó có: (1) HTML định nghĩa nội dung của trang Web, (2) CSS xác định cách thức mà nội dung của trang Web được hiển thị và (3) JavaScript để lập trình các hành vi của các phần tử trên trang Web. JavaScript ngày càng phổ biến trong phát triển ứng dụng. Ngày nay ngoài các ứng dụng Web, nhiều ứng dụng trên máy tính, máy chủ và thiết bị di động cũng sử dụng JavaScript.

JavaScript có thể làm được nhiều việc Thay đổi nội dung phần tử HTML. Ví dụ có phần tử p như sau:

```
<h4 class="contact-title" style>Hòm thư</h4>
<a href="mailto:(Webmail@gmail.com)" id="info-email" style="display: none"></a>
```

Có thể sử dụng mã JavaScript sau để thay đổi nội dung của thẻ HTML:

```
let emailElement = document.getElementById("info-email")
emailElement.innerText = "phamduongminh@gmail.com";
emailElement.href= mailto:(phamduongminh@gmail.com)
emailElement.style.display= "block"
```

Một trong những hàm JavaScript thông dụng là **getElementById()**, khi sử dụng hàm này JavaScript sẽ tìm các phần tử theo id trên tài liệu HTML, từ đó có thể lấy nội dung cũng như thay đổi nội dung của các phần tử này. Ví dụ trên, phần tử **<a>** có id là **"info-email"** được tìm thấy và thay nội dung hiển thị cũng như thuộc tính **href** của phần tử này. Kết quả:

Hình 5-1 Phần tử a sau khi thay đổi nội dung và thuộc tính bằng Javascript

Có thể thấy Javascript có thể thay đổi nội dung của thẻ **HTML**, ví dụ như trường hợp trên đưa nội dung **phamduongminh@gmail.com** vào trong phần tử **<a>.** Ngoài ra, các thuộc tính và định dạng (style) của phần tử HTML cũng có thể thay đổi. Như ví dụ trên thuộc tính href cũng như định dạng của thẻ (thuộc tính style) đã được thay đổi bằng mã JavaScript.

5.1.2 Chèn JavaScript vào tài liệu HTML

Để chèn mã JavaScript vào HTML sử dụng thẻ <**script**>. Phần tử này sẽ khai báo một kịch bản **JavaScript** bằng việc viết mã trực tiếp trong phần tử <script> này. Ví dụ:

```
<script>
    let emailElement = document.getElementById("info-email")
    emailElement.innerText = "phamduongminh@gmail.com";
    emailElement.href= "mailto:(phamduongminh@gmail.com)"
    emailElement.style.display= "block"
</script>
```

Các thẻ <**script**> này có thể đặt ở bất kỳ vị trí nào trong file HTML. Thông thường các đoạn mã sẽ được lưu trong các file có phần mở rộng là .js. Khi sử dụng file JavaScript này cần tham chiếu đến thông qua thuộc tính **src** của thẻ <**script**>:

```
<script src="js/main.js"></script>
```

Sử dụng các file JavaScript có nhiều lợi thế, nó giúp tách mã JavaScript và mã HTML. Điều này khiến các đoạn mã dễ đọc và bảo trì. Ngoài ra file JavaScript thường được lưu trong bộ đệm khiến việc tải trang được nhanh hơn trong những lần tải sau.

Tuy nhiên cần lưu ý rằng có những trình duyệt sẽ không hỗ trợ chạy các kịch bản JavaScript. Khi đó tài liệu HTML có thể hiển thị không đúng như ý đồ của người thiết kế. Trong trường hợp này, để dự phòng cần sử dụng thêm phần tử <**noscript**>. Phần tử này sẽ hiển thị một thông báo khi trình duyệt không hỗ trợ chạy kịch bản. Ví dụ trình duyệt sẽ hiển thị chữ "Không hỗ trợ Javascript" nếu nó không thể chạy đoạn kịch bản được khai báo trong phần tử <**script**>:

```
<script>
    let emailElement = document.getElementById("info-email")
    emailElement.innerText = "phamduongminh@gmail.com";
    emailElement.href= "mailto:(phamduongminh@gmail.com)"
    emailElement.style.display= "block"
</script>
<noscript>Không hỗ trợ JavaScript</noscript>
```

5.1.3 Hiển thị log

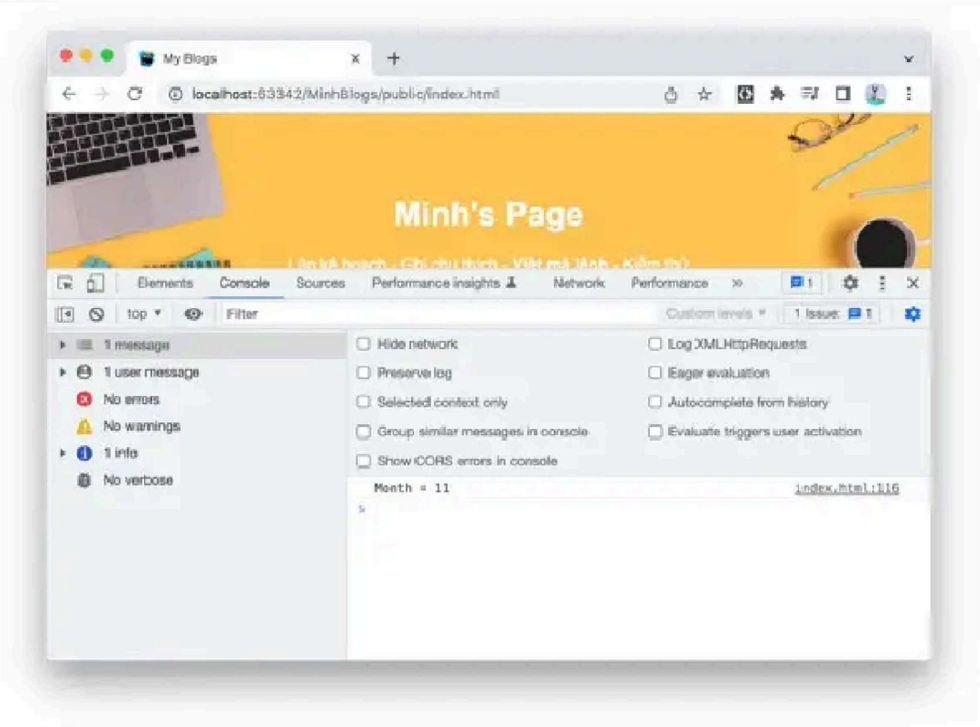
Trong quá trình lập trình muốn biết giá trị của một biến hay biểu thức tại một thời điểm chạy có thể sử dụng phương thức console.log(). Khi đó giá trị cần in ra sẽ hiển thị trong cửa sổ log. Xét ví dụ trường hợp muốn in ra biến month:

```
<script>
    const date = new Date('December 25, 2023');
    const month = date.getMonth();
    console.log("Month = " + month);
</script>
```

Bên cạnh đó, nhà phát triển có thể sử dụng các cách khác nhau để xuất thông tin và hiển thị thông tin đó trên trình duyệt ví dụ sử dụng thay đổi nội dung của phần tử HML bằng việc thay đổi thuộc tính **innerText** hoặc **innerHTML**, sử dụng hàm **document.write()**, hay hiển thị các thông báo bằng phương thức alert(). Khi sử dụng **innerHTML**, người lập trình thông thường

sẽ sử dụng phương thức **document.getElementById(id)** để lấy phần tử HTML trước sau đó thay đổi nội dung thông qua **innerText** hoặc **innerHTML**. Hàm **document.write()** phù hợp cho mục đích test. Cần lưu ý: nếu sử dụng document.write() sau khi trang HTML được tải, sẽ xoá toàn bộ các nội dung đang có trên file HTML. Sử dụng **alert()** để hiện thị một hộp thoại.

Tuy nhiên cách phổ biến và dễ dàng hơn cả vẫn là sử dụng phương thức **console.log()** như trên. Đối với trình duyệt **Chrome**, có thể bấm **F12** để để bật công cụ **Developer Tools** và vào tab Console để xem kết quả in ra từ câu lệnh log. Ví dụ với mã lệnh như trên, kết quả hiển thị ra cửa sổ Log là:



Hình 5-2 Kết quả hiển thị Log trên cửa sổ Console

5.1.4 Ghi chú thích

Để viết các chú thích trong khi lập trình JavaScript có thể sử dụng // hoặc /* */. Trong đó để ghi chú thích trên một dòng sử dụng //. Trường hợp cần ghi chú thích trên nhiều dòng, sử dụng /* để bắt đầu và kết thúc bằng */. Ngoài tác dụng ghi chú thích, sử dụng // hoặc /* */ với các khối lệnh để ngăn việc thực thi các khối lệnh đó. Ví dụ ghi chú thích trên 1 dòng:

```
<script>
    const date = new Date('December 25, 2023');
    const month = date.getMonth();  //Lấy ra tháng của ngày Giáng sinh
    console.log("Month = " + month);  //Hiển thị ra cửa sổ Console
</script>
```

Để ghi chú thích trên nhiều dòng dùng /* và */ như sau:

```
<script>
    /*
     Đưa ra lời chúc mừng phù hợp thơi điểm
     - Tháng 11: Chúc mừng ngày nhà gia
     - Tháng 12: Chúc mừng lễ giáng sinh
     (Chú ý với hàm getMonth() trả về giá trị từ 0-11 tương ứng tháng 1-12)
     */
    const date = new Date();
    const myquote = document.getElementById("myquote")
    if (date.getMonth() == 10) {
        myquote.innerHTML = "Mừng ngày <b>nhà giáo Việt Nam</b>"
    } else if (date.getMonth() == 11) {
        myquote.innerHTML = "Chúc <b>giáng sinh</b> vui vẻ"
    }
</script>
```

5.1.5 Từ khóa, từ dành riêng

Trong mỗi một ngôn ngữ lập trình **"từ khóa" (keywords)** hay **"từ dành riêng" (reserved words)** là tập hợp những từ có ý nghĩa đặc biệt trong chương trình. Khi học bất kỳ ngôn ngữ lập trình nào cũng cần biết cách sử dụng của các từ khóa này. Đồng thời lưu ý không được sử dụng các từ này để định danh (đặt tên) cho các khai báo mới trong chương trình như biến, phương thức, lớp, đối tượng ... Các "từ khóa", "từ dành riêng" trong JavaScript bao gồm:

Bảng 5-1 Danh sách từ khóa trong JavaScript

abstract	arguments	await*	boolean
break	byte	case	catch
char	class*	const	continue
debugger	default	delete	do
double	else	enum*	eval
export*	extends*	false	final
finally	float	for	function
goto	if	implements	import*
in	instanceof	int	interface
let*	long	native	new
null	package	private	protected
public	return	short	static
super*	switch	synchronized	this
throw	throws	transient	true
try	typeof	var	void
volatile	while	with	yield

Trong JavaScript, không thể sử dụng các từ dành riêng này làm biến, nhãn hoặc tên hàm. Ngoài ra cũng nên tránh sử dụng tên của các đối tượng, thuộc tính và phương thức được tích hợp sẵn JavaScript như **Array, Dath, Math, valueOf()**… Tránh sử dụng tên của các đối tượng và thuộc tính HTML và Window như **button, confirm, history** …Tránh sử dụng tên của tất cả các trình xử lý sự kiện HTML: **onblur, onclick, onmouseover,** …

5.1.6 Thực thi mã Javascript bằng nodejs

Thực thi mã JavaScript bằng Node.js là quá trình chạy mã JavaScript trên máy tính bằng Node.js runtime environment (Thay vì bằng trình duyệt). Node.js cho phép sử dụng JavaScript để phát triển các ứng dụng phía máy chủ, các ứng dụng dòng lệnh và các công cụ khác. Để thực thi mã JavaScript bằng Node.js, cần tạo một file JavaScript và sử dụng lệnh "node tên_file.js" để thực thi mã đó. Khi thực thi, Node.js sẽ tải và biên dịch mã JavaScript và thực thi nó trên máy tính. Việc lập trình nodejs để viết các kịch bản phía máy chủ nằm ngoài phạm vi của giáo trình này.

5.2 SỬ DỤNG BIẾN

5.2.1 Khai báo biến

Trong một ngôn ngữ lập trình, các biến được sử dụng để lưu trữ các giá trị dữ liệu. JavaScript sử dụng từ khóa **var** để khai báo các biến. Ví dụ:

```
var a = 21;
var b = -6;
var c = a + b;
```

Khi khai báo biến trong JavaScript, kiểu dữ liệu là động. Điều đó đồng nghĩa với việc một biến có thể lưu trữ nhiều kiểu dữ liệu tại các thời điểm khác nhau. Ví dụ một biến có thể nhận giá trị là số, hoặc giá trị là chuỗi ký tự tại các thời điểm khác nhau.

Việc đặt tên biến cần theo quy tắc:

- Không đặt tên biến trùng với **từ khoá, từ dành riêng** đã nêu ở trên.
- Tên biến có thể chứa **số**, **chữ cái, gạch dưới** (_), hoặc **ký tự đô la** ($).
- **Ký tự đầu tiên** phải là chữ cái, gạch dưới (_), hoặc ký tự đô la ($).
- JavaScript **phân biệt** chữ hoa và chữ thường.

Những biến như sau đặt tên hợp lệ:

```
var ten = "Phạm Dương Minh"
var nam_sinh = 2000
var $tuoi = 18
var QUOCTICH1 = "Vietnam"
```

Các khai báo biến sau sử dụng tên không hợp lệ:

```
var email@ = "phamduongminh@gmail.com" //Có chứa ký tự đặc biệt
var 3diem = 8 //Bắt đầu bởi chữ số
var noi sinh = "Ha noi"  //Tên biến có chứa khoảng trắng
var class = "Cong nghe thong tin"  //Tên biến trùng với từ khóa class
```

Chú ý JavaScript phân biệt chữ hoa và chữ thường, **name** hay **Name** được xem là 2 biến khác nhau. Sau khi biến được khai báo có thể gán giá trị cho biến bằng cách sử dụng **toán tử gán** (=). Việc gán giá trị có thể thực hiện ngay khi khai báo biến hoặc sau khi khai báo biến. Trong trường hợp khai báo lại một biến đã khai báo và có giá trị trước đó, biến đó sẽ không bị mất giá trị, ví dụ:

```
var name = "Phạm Dương Minh";
var name;
```

Sau khi khai báo lại biến **name**, giá trị của biến đó vẫn là "Phạm Dương Minh"

5.2.2 Phạm vi biến

Phạm vi (Scope) xác định khả năng truy cập (khả năng hiển thị) của các biến. Trong JavaScript có hai loại phạm vi:

- Phạm vi **cục bộ (local).**
- Phạm vi **toàn cục (global).**

Biến cục bộ hay còn được gọi là **biến địa phương** là các biến được khai báo trong một hàm và sẽ có **phạm vi cục bộ trong hàm đó**. Ví dụ các biến được khai báo trong một hàm một hàm sẽ trở thành biến cục bộ có phạm vi hàm. Tất nhiên, chúng chỉ có thể được truy cập từ bên trong hàm. Ví dụ:

```
function myInfo() {
   var name = "Phạm Dương Minh";
   //Biến name là biến cục bộ của hàm showInfo()
}
//Biến name không thể được truy cập tại đây
```

Biến cục bộ được tạo khi hàm bắt đầu và bị xóa khi hàm hoàn thành. Vì các biến cục bộ chỉ được nhận dạng bên trong các hàm của chúng, các biến có cùng tên có thể được sử dụng trong các hàm khác nhau.

Một biến được khai báo bên ngoài một hàm, trở thành **biến toàn cục** và nó có **phạm vi toàn cục**. Điều đó có nghĩa là mọi phương thức có thể truy cập biến đó. Ví dụ:

```
var school_name = "Kinh te quoc dan";  //Biến toàn cục school_name
// tại đây có thể truy cập biến school_name
function myInfo () {
   // tại đây cũng có thể truy cập biến school_name
}
```

Tuy nhiên, cần lưu ý rằng, nếu trong một hàm gán một giá trị cho một **biến chưa được khai báo**, biến đó sẽ **tự động trở thành biến toàn cục**. Ví dụ mã này sẽ khai báo một biến toàn cục school_name:

```
myInfo();
// tại đây có thể sử dụng biến school_name (sau khi gọi hàm myInfo())
function myInfo() {
 school_name = "Kinh te quoc dan";
}
```

Trong JavaScript còn cho phép khai báo biến sử dụng từ khóa **let**. Khai báo **let** và **var** khác nhau về phạm vi truy cập. Xét ví dụ như sau:

```
{
 var a = 2;
}
// Có thể truy cập biến a ở đây
{
 let b = 2;
```

```
}
// Không thể truy cập biến b ở đây
```

Trong chương trình có nhiều khối lệnh được hình thành. Khi đó các biến khai báo trong khối lệnh sử dụng **var** thì vẫn có thể truy cập bên ngoài khối lệnh. Đây là một điều rất thuận lợi tuy nhiên lại gây ra sự nhập nhằng khi sử dụng biến. Tuy nhiên nếu khai báo biến trong khối lệnh sử dụng từ khóa **let** thì biến đó phạm vi truy cập chỉ là trong khối lệnh đó mà thôi. Khi khối lệnh kết thúc biến đó sẽ tự động bị hủy. Xét một ví dụ khác:

```
var a = 0;
for (var a = 0; a < 10; a++) {

}
//a có giá trị bằng 10

let b = 0;
for (let i = 0; i < 10; i++) {
 // some statements
}
// b có giá trị bằng 0
```

Trong ví dụ trên, sau khi vòng lặp **for** kết thúc, a có giá trị bằng 10 trong khi b lại có giá trị = 5. Bản chất là vì khi sử dụng **var**, chỉ có 1 biến **a** duy nhất. Trong khi nếu khai báo biến sử dụng **let** thì đó là **2 biến khác nhau**. Giá trị biến **b** được sử dụng và thay đổi trong vòng lặp **for** sẽ bị hủy sau khi vòng lặp này kết thúc.

Ngoài ra, từ khóa **const** được sử dụng để khai báo **hằng số**, một biến **const** bắt buộc phải gán giá trị khi khai báo, và không thể thay đổi giá trị sau khi gán. Ví dụ khai báo hằng số PI như sau:

```
const PI = 3.141592653589793;
PI = 3.141592; // Không hợp lệ vì không thể thay đổi giá trị của PI
```

5.2.3 Cẩu - Hoisting

Cẩu (**Hoisting**) là hành vi mặc định của JavaScript về việc **di chuyển các khai báo lên đầu**. Khác với nhiều ngôn ngữ lập trình khác, trong JavaScript, một biến có thể được khai báo sau khi nó được sử dụng. Điều đó là hợp lệ vì JavaScript di chuyển tất cả các khai báo lên trên cùng phạm vi của biến đó (như lên đầu của hàm với biến cục bộ). Tuy nhiên việc di chuyển này chỉ áp dụng với các khai báo bằng từ khóa **var,** và chỉ áp dụng cho các khai báo biến không đi kèm với khởi tạo giá trị. Ví dụ việc viết mã sau là hoàn toàn hợp lệ:

```
tuoi = 18
ten = "Phạm Dương Minh"
var tuoi
var ten
```

5.3 KIỂU DỮ LIỆU

Kiểu dữ liệu (data type) của một biến có thể là: số, chuỗi ký tự, noolean, đối tượng hoặc kiểu rỗng (null).

5.3.1 Kiểu số

JavaScript chỉ có một kiểu dữ liệu số. Một số có thể có phần thập phân hoặc không. Ví dụ:

```
var tuoi = 28  //Số không có phần thập phân
var hesoluong = 2.34 //Số có phần thập phân
var luongcoban = 1.5e6 //Số biểu diễn theo dạng mũ e (tương đương 1500000)
var saiso = 1e-3 //Tương đương với 1/1000 hay 0.001
```

Khi số được lưu trữ ở dạng chuỗi và cần thực hiện các phép tính JavaScript sẽ cố gắng thực hiện chuyển từ chuỗi thành số khi sử dụng toán tử toán học trên các chuỗi số đó. Tuy nhiên điều này chỉ đúng với các toán tử **nhân** (*), **chia** (/), hoặc **trừ** (-). Ví dụ:

```
var hesoluong = "2.34"
var luongcoban = "1.5e6"
console.log(luongcoban * hesoluong)  //3510000
```

Ví dụ trên vẫn cho ra kết quả chính xác là **3510000.** Tuy nhiên nếu thay **toán tử nhân** (*) bằng **toán tử cộng** (+) thì nó lại trở thành toán tử ghép chuỗi và kết quả in ra sẽ là **"1.5e62.34".**

Trong trường hợp không thể thực hiện được phép tính kết quả trả về NaN (Not a Number). Ví dụ:

```
var hesoluong = "2.34"
var luongcoban = "1.5a6"
console.log(luongcoban * hesoluong)  //NaN
```

Không giống như hầu hết ngôn ngữ lập trình khác, JavaScript không định nghĩa các kiểu dữ liệu số khác nhau. Tất cả các số trong JavaScript đều là số thực và sử dụng **64 bit.** Với 64 bit một số thực trong JavaScript sẽ sử dụng **52 bits cho phần giá trị**, **11 bit cho phần số mũ** và **1 bit cho dấu**. Độ chính xác của số trong JavaScript là 15 chữ số. Ví dụ:

```
var so = 12345678901234567890123.1234567890
console.log(so) //Giá trị in ra của so sẽ là 1.2345678901234568e+22
```

Các số trong JavaScript cũng thể nhận giá trị **vô cùng** (Infinity) hoặc **âm vô cùng** -Infinity trong trường hợp số đó vượt quá khoảng lưu trữ số của JavaScript. Ví dụ:

```
var so = 3
console.log(so/0) // Infinity
```

Số trong JavaScript cũng có thể khai báo số như là một đối tượng với từ khóa **new** như sau:

```
var tuoi = new Number(20);
```

Hàm **Number()** cũng được sử dụng để chuyển nhiều kiểu dữ liệu khác về số. Ví dụ:

```
Number(true);   // 1
Number(false);   // 0
Number("20");   // 20
Number("   20");  // 20
Number(" 20   ");  // 20
Number("20.5");   // 20.5
Number("Hai muoi");  // NaN
```

Bên cạnh hàm **Number()** thì hàm **parseInt()** và **parseFloat()** cũng được sử dụng để chuyển đổi chuỗi thành các số **int (số nguyên)** hoặc **float (số thực)** tương ứng.

Các số có các phương thức có thể sử dụng như toExponential(), toFixed(), toPrecision()... Ví dụ:

```
var gia = 180.25;
console.log(gia.toExponential()) //1.8e+7
console.log(gia.toFixed()) //18000000
console.log(gia.toPrecision()) //18000000
```

Number có nhiều phương thức hữu ích để làm việc với số, ngoài ra nó còn có các hằng số quan trọng. Ví dụ:

```
console.log(Number.EPSILON) //2.220446049250313e-16 : Hiệu của 1 và số nhỏ nhất
lớn hơn 1
console.log(Number.MIN_VALUE) //5e-324 : Số nhỏ nhất
console.log(Number.MAX_VALUE) //1.7976931348623157e+308 : Số lớn nhất
console.log(Number.MAX_SAFE_INTEGER) //9007199254740991 : Số nguyên "an toàn"
lớn nhất
console.log(Number.MIN_SAFE_INTEGER) //-9007199254740991 : Số nguyên "an toàn"
nhỏ nhất
console.log(Number.NEGATIVE_INFINITY) //-Infinity : Âm vô cùng
console.log(Number.POSITIVE_INFINITY) //Infinity : Dương cô cùng
```

Các số "an toàn" là các số mà hệ thống đảm bảo việc tính toán chính xác, một kết quả tính toàn nằm ngoài khoảng "an toàn" có thể không đảm bảo sự chính xác.

5.3.2 Kiểu chuỗi ký tự

Chuỗi để lưu trữ và thao tác với ký tự và văn bản. Một chuỗi được đặt trong các dấu nháy kép " " hoặc nháy đơn ' '. Ví dụ:

```
var ten = "Phạm Dương Minh";
var lop = "Công nghệ thông tin";
```

Có thể để các dấu nháy kép, và nháy đơn bên trong chuỗi. Dấu nháy kép có thể tồn tại trong mọt chuỗi khai báo bằng dấu nháy đơn và ngược lại mà không cần chú ý điều gì. Một cách khác nữa là sử dụng dấu \. Vì vậy, các trường hợp sau đều hợp lệ:

```
var chuoi1 = "Minh cho rằng: 'Lập trình rất đơn giản'";
var chuoi2 = 'Minh cho rằng: "Lập trình rất đơn giản"';
var chuoi3 = "Minh cho rằng: \"Lập trình rất đơn giản\"";
var chuoi4 = 'Minh cho rằng: \'Lập trình rất đơn giản\'';
```

Ký tự \ còn giúp hiển thị các ký tự đặc biệt khác/Ví dụ:

```
console.log("1. Lập trình\b Web") //ký tự xóa
console.log("2. Lập trình\f Web") //Xuống dòng
console.log("3. Lập trình\n Web") //Xuống dòng về đầu dòng
console.log("4. Lập trình\r Web") // Về đầu dòng
console.log("5. Lập trình\t Web") //Ký tự tab
console.log("6. Lập trình\\ Web") // Ký tự \
```

Kết quả các dòng lệnh trên sẽ như sau:

```
1. Lập trìn Web
2. Lập trình
```

```
             Web
3. Lập trình
 Web
 Webập trình
5. Lập trình     Web
6. Lập trình\ Web
```

Độ dài của chuỗi được xác định thông qua thuộc tính length. Ví dụ:

```
var quoc_gia = "Việt Nam";
console.log(quoc_gia.length)  //8
```

Một chuỗi có thể một đối tượng, chính vì vậy để tạo một chuỗi có thể sử dụng một trong 2 cú pháp như ví dụ phía dưới. Tuy nhiên, cần lưu ý khi sử dụng phép so sánh (x===y) thì kết quả trả về sẽ là **false** do 2 biến khác nhau về kiểu dữ liệu. Để so sánh giá trị 2, dùng toán tử ==. Cụ thể như sau:

```
var name1= "Minh";  //String
var name2= new String("Minh");  //Object
console.log(name1===name2) //false
console.log(name1==name2)  //true
```

Javascript cung cấp rất nhiều phương thức để làm việc với chuỗi. Ví dụ để tìm một chuỗi trong một chuỗi có thể sử dụng phương thức **indexOf()**. Ví dụ:

```
var khauhieu = "Lập kế hoạch - Ghi chú thích - Viết mã lệnh - Kiểm thử"
console.log(khauhieu.indexOf("-"))  //Trả về 13
console.log(khauhieu.indexOf("JavaScript"))  //Trả về -1
```

Ví dụ hàm **indexOf()** trả về -1 trong trường hợp không tìm thấy, hàm trên cũng có thể truyền thêm tham số thứ 2 để xác định vị trí bắt đầu tìm kiếm. Trường hợp muốn tìm ngược từ dưới có thể sử dụng phương thức **lastIndexOf()**. Ví dụ:

```
var khauhieu = "Lập kế hoạch - Ghi chú thích - Viết mã lệnh - Kiểm thử"
console.log(khauhieu.lastIndexOf("-"))  //Trả về 44
```

Để thay thế nội dung của các ký tự trong chuỗi có thể sử dụng phương thức **replace()**. Lưu ý phương thức **replace()** không thay đổi chuỗi ban đầu mà sẽ trả về chuỗi mới có nội dung đã thay đổi, ví dụ:

```
var khauhieu = "Lập kế hoạch - Ghi chú thích - Viết mã lệnh - Kiểm thử"
var khauhieu2 = khauhieu.replace("Kiểm thử", "Vận hành");
console.log(khauhieu) //Lập kế hoạch - Ghi chú thích - Viết mã lệnh - Kiểm thử
console.log(khauhieu2) //Lập kế hoạch - Ghi chú thích - Viết mã lệnh - Vận hành
```

Phương thức **concat()** được sử dụng để nối 2 chuỗi ký tự. Ví dụ:

```
var khauhieu = "Lập kế hoạch - Ghi chú thích - Viết mã lệnh - Kiểm thử"
var khauhieu2 = khauhieu.concat(" - Vận hành");
console.log(khauhieu2)
```

Phương thức trim() được sử dụng để xóa các khoảng trắng ở đầu và cuối chuỗi.

```
var language = "  JavaScript  ";
language = language.trim(); // Javascript
```

Chuyển chuỗi về một mảng, có thể sử dụng phương thức **split().** Phương thức này sẽ phân tách chuỗi ban đầu theo một ký hiệu nào đó, ví du:

```
var khauhieu = "Lập kế hoạch - Ghi chú thích - Viết mã lệnh - Kiểm thử"
var arr = khauhieu.split("-")
var arr = arr.map(s => s.trim()) //Duyệt và trim() các phần tử trong mảng
console.log(arr) // ['Lập kế hoạch','Ghi chú thích','Viết mã lệnh','Kiểm thử']
```

Phương thức toUpperCase() để chuyển từ thường sang chữ hoa, hay phương thức toLowerCase() để chuyển từ chữ hoa sang chữ thường.

```
var khauhieu = "JavaScript"
console.log(khauhieu.toUpperCase()) //JAVASCRIPT
console.log(khauhieu.toLowerCase()) //javascript
```

Bên cạnh các phương thức kể trên Javascript còn cung cấp rất nhiều phương thức khác làm việc với chuỗi, ví dụ: slice(), substring(), substr(), padStart(), padEnd(), charAt()...

5.3.3 Kiểu boolean

Một biến có kiểu **boolean** thì giá trị chỉ có thể nhận giá trị **true** hoặc **false**. Kiểu boolean thường được sử dụng trong các biểu thức điều kiện. Ví dụ:

```
let lapgiadinh = false;
if(lapgiadinh){
    console.log("Đã lập gia đình")
}
```

Kiểu boolean cũng có thể được biểu diễn dưới dạng kiểu Object bằng cách sử dụng lớp Boolean. Ví dụ:

```
let lapgiadinh = new Boolean(false);
```

Tương tự như việc dùng String và Number, kiểu của biến boolean khi khai báo bằng lớp Boolean sẽ có kiểu là Object. Trong hầu hết các trường trường hợp thì việc sử dụng một đối tượng Boolean là không cần thiết. Sử dụng lớp Boolean khiến mã lện dài hơn, chương hoạt động chậm hơn. Vì vậy cần cân nhắc khi sử dụng.

5.3.4 Kiểu đối tượng

Một biến trong Javascript có thể khai báo là một đối tượng. Với kiểu đối tượng, một đối tượng trong JavaScript được đặt trong **dấu ngoặc nhọn "{}"**, Các thuộc tính của đối tượng được viết theo dạng cặp phân cách bởi **dấu hai chấm ":"**. Các thuộc tính được phân cách bởi **dấu phẩy ","**. Ví dụ một đối tượng mô tả một sinh viên trong lớp:

```
var sinhvien = {
    ho: "Pham",
    dem: "Duong",
    ten: "Minh",
    tuoi: 21,
    quequan: "Hai Duong",
    lop: "Cong nghe thong tin 62"
};
```

Trong JavaScript cũng có thể tạo đối tượng bằng cách sử dụng từ khóa **new**. Các biến đối tượng tạo ra có thể từ các lớp tự định nghĩa hoặc các lớp có sẵn. Ví dụ tạo ra một đối tượng ngày tháng từ lớp Date:

```
var ngaysinh = new Date("25/12/2000");
```

Kiểu dữ liệu đối tượng gắn liền với kỹ thuật lập trình hướng đối tượng, sẽ được trình bày kỹ hơn trong phần sau của giáo trình.

5.3.5 Kiểu rỗng null

Trong JavaScript, kiểu **rỗng (null)** có nghĩa là "không có gì". Nó thường được dùng để ám chỉ **dữ liệu rỗng** hoặc **không còn lưu trữ** gì bên trong. Ví dụ:

```
var sinhvien1 = {
    ho: "Pham",
    dem: "Duong",
    ten: "Minh",
    tuoi: 21,
    quequan: "Hai Duong",
    lop: "Cong nghe thong tin 62"
};
sinhvien1 = null;
console.log(sinhvien1) //Giá trị của sinh viên là null nhưng kiểu vẫn là Object
```

Kiểu của null là Object, trong khi một biến chưa được khởi tạo giá trị sẽ có kiểu là **undefined:**

```
var sinhvien1
console.log(sinhvien1) //undefined
```

5.3.6 Kiểm tra kiểu

Để **kiểm tra kiểu** của một biến tại một thời điểm, có thể sử dụng toán tử **typeof**, ví dụ:

```
typeof true      // Trả về "boolean"
typeof {}        // Trả về "object"
typeof "Minh"    // Trả về "string"
typeof 21        // Trả về "number"
typeof x        //  Trả về "undefined" nếu x là một biến chưa khai báo
```

Đối với đối tượng, muốn kiểm tra một biến có phải là thể hiện của một lớp hay không, sử dụng toán tử **instanceof**. Ví dụ:

```
function SinhVien(ten, tuoi) {
    this.ten = ten;
    this.tuoi = tuoi;
}
const sinhvien = new SinhVien('Dương Minh', 21);

console.log(typeof sinhvien);  //object
console.log(sinhvien instanceof String);  //false
console.log(sinhvien instanceof SinhVien);  //true
```

5.3.7 Chuyển kiểu

Các biến JavaScript có thể được chuyển đổi thành kiểu dữ liệu khác, mà vẫn có thể "giữ" giá trị. Ví dụ phương thức String() có thể **chuyển đổi số thành chuỗi:**

```
console.log(String(21)) //Chuyển một số thành chuỗi, kết quả "21"
console.log(String({ten:"Minh",tuoi:21})) //Chuyển một đối tượng thành chuỗi,
kết quả "[object Object]"
console.log(String(new Date())) //Chuyển một đối tượng thành chuỗi kết quả: "Sat
Nov 12 2022 18:01:50 GMT+0700 (Indochina Time)"
console.log(String(true)) //Chuyển một boolean thành chuỗi, kết quả "true"
console.log(String(null)) //Chuyển một giá trị rỗng thành chuỗi, kết quả "null"
```

Phương thức **toString()** cũng thực hiện điều tương tự, tuy nhiên toString() chỉ có thể gọi từ các biến, và không thể gọi từ một biến **null** hoặc **undefined.**

```
let x = 21;
console.log(x.toString())  //Chuyển một số thành chuỗi, kết quả "21"
x = {ten: "Minh", tuoi: 21}
console.log(x.toString())  //Chuyển một đối tượng thành chuỗi, kết quả "[object
Object]"
x = new Date()
console.log(x.toString())  //Chuyển một đối tượng thành chuỗi, kết quả "Sat Nov
12 2022 18:01:50 GMT+0700 (Indochina Time)"
x = true
console.log(x.toString()) //Chuyển một boolean thành chuỗi, kết quả "true"
x = null
console.log(x.toString())  //Không thể thực hiện được
```

Phương thức **Number()**, **Boolean**() cũng có tác dụng tương tự phương thức String(). Chúng cũng chuyển các kiểu dữ liệu khác nhau thành kiểu **number** và **boolean.** Bên cạnh đó, JavaScript cũng cung cấp nhiều phương thức để chuyển kiểu như **parseInt(), parseFloat()**. Tuy nhiên cần lưu ý rằng việc chuyển kiểu có thể phát sinh lỗi khi dữ liệu không thể chuyển được, vì vậy cần xử lý lỗi (ngoại lệ) để đảm bảo chương trình hoạt động ổn định.

5.4 TOÁN TỬ

Toán tử là thành phần không thể thiếu trong một biểu thức, nó kết hợp với các toán hạng để hình thành lên một biểu thức nhằm đáp ứng nhiệm vụ tính toán của một chương trình. Trong JavaScript có nhiều loại **toán tử**:

- Toán tử gán.
- Toán tử toán học.
- Toán tử với chuỗi.
- Toán tử so sánh.
- Toán tử logic.
- Toán tử kiểm tra kiểu.
- Toán tử bit.

5.4.1 Toán tử gán

Toán tử **gán** được sử dụng để gán giá trị cho một biến:

Toán tử	Ví dụ	Tương đương với
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y

5.4.2 Toán tử toán học

Toán tử toán học được sử dụng để thực hiện các phép tính toán học đơn giản trên các toán hạng của một biểu thức:

Toán tử	Mô tả
+	Cộng
-	Trừ
*	Nhân
**	Luỹ thừa
/	Chia
%	Chia lấy phần dư
++	Tăng một đơn vị
--	Giảm một đơn vị

5.4.3 Toán tử so sánh

Toán tử so sánh được sử dụng để so sánh hai toán hạng:

Toán tử	Mô tả
==	Bằng giá trị
===	Bằng giá trị và cùng kiểu dữ liệu
!=	Khác giá trị
!==	Không bằng giá trị hoặc kiểu dữ liệu
>	Lớn hơn
<	Nhỏ hơn
>=	Lớn hơn hoặc bằng
<=	Nhỏ hơn hoặc bằng

5.4.4 Toán tử bitwise

Toán tử bitwise làm việc với các số nhị phân. Với số thuộc hệ khác, các toán hạng sẽ được chuyển sang hệ nhị phân trước khi thực hiện tính toán.

Toán tử	Mô tả	Ví dụ	Tương đương với	Kết quả
&	AND	5 & 1	0101 & 0001	0001
|	OR	5 | 1	0101 | 0001	0101
~	NOT	~ 5	~0101	1010
^	XOR	5 ^ 1	0101 ^ 0001	0100

<<	Dịch trái	5 << 1	0101 << 1	1010
>>	Dịch phải	5 >> 1	0101 >> 1	0010

5.5 HÀM

5.5.1 Khai báo và sử dụng hàm

Một hàm là một khối mã lệnh được thiết kế để thực hiện một công việc cụ thể nào đó. Một hàm trong JavaScript được khai báo theo cách truyền thống bằng từ khoá **function**, tiếp theo là tên và dấu () chứa các tham số.

```
function tinhtong(a, b) {
 return a + b;   // Giá trị trả về là tổng của a và b
}
var tong = tinhtong(5, 10); // Gọi hàm, giá trị trả về lưu vào biến tong
```

Các tham số truyền vào hàm, được đặt trong dấu () và được truyền vào khi gọi hàm. Một hàm có giá trị trả về thông qua lệnh **return** . Khi gặp lệnh **return** hàm cũng kết thúc việc thực thi và giá trị sẽ được trả về nơi hàm được gọi. Lưu ý các biến được khai báo trong hàm được gọi là các **biến địa phương (biến local)**, và chỉ có thể được truy cập bên trong của hàm. Các biến này được tạo và sử dụng khi hàm bắt đầu thực thi, và sẽ được xoá khi hàm kết thúc.

5.5.2 Hàm mũi tên

Bên cạnh việc khai báo hàm theo cách truyền thống bằng từ khóa **function**. Trong đặc tả ES6 (ECMAScript 6), cho phép sử dụng hàm mũi tên. Ví dụ một hàm khai báo với từ khóa function():

```
function tinhtong(a, b) {
    return a + b;
}
```

Chuyển thành hàm mũi tên như sau:

```
let tinhtong = (a,b) => {
    return a + b;
}
```

Vì hàm trên chỉ có một câu lệnh trong phần thân, vì vậy có thể viết:

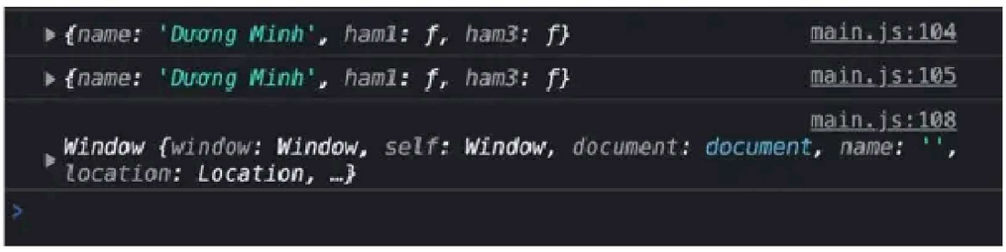
```
let tinhtong = (a,b) => a + b;
```

Có thể thấy, trong một số trương hàm mũi tên cho phép khai báo một số hàm với cú pháp ngắn hơn. Tuy nhiên, cần lưu ý khi sử dụng **this** đối với hàm mũi tên. Đối với hàm khai báo bằng **function** thì **this** là đối tượng gọi hàm. Còn đối với hàm mũi tên, **this** là đối tượng định nghĩa ra hàm. Ví dụ:

```
const sv1 = {
    name: "Dương Minh",
    ham1: function () {
        console.log(this)
        let ham2 = () => console.log(this)
        ham2()
```

```
    },
    ham3: () => console.log(this)
};
sv1.ham1();
sv1.ham3();
```

Trường hợp này **this** trong ham1() và ham2() chính là đối tượng **sv1**. Còn this trong ham3() sẽ phụ thuộc việc **ham3()** được định nghĩa ở đâu. Nếu từ trình duyệt thì **this** đó sẽ là đối tượng window. Kết quả in ra cửa sổ Console của Developer Tools của trình duyệt như sau:



5.5.3 Các hàm của các đối tượng có sẵn

JavaScript cung cấp nhiều lớp trong đó có có sẵn các hàm. Các hàm này rất hữu ích trong quá trình lập trình. Ví dụ như lớp **Math** cung cấp rất nhiều các hàm toán học:

```
Math.round(9.1); // Trả về giá trị làm tròn của 9.1 là 9
Math.pow(2, 3);  // Trả về hai mũ 3 là 8
Math.sqrt(9);  // Trả về căn của 9 là 3
Math.ceil(9.4);  // Trả về 10
Math.random(); //Trả về một giá trị ngẫu nhiên
```

Lớp **Array** cung cấp nhiều phương thức để làm việc với mảng, ví dụ:

```
Array.of(1, 2, 3)  //Hàm of() tạo mảng từ các giá trị
Array.from(new Set(['Công', 'Nghệ', 'Thông', 'Tin']))  //Hàm from() tạo mảng từ
một đối tượng
```

Ngoài ra còn rất nhiều lớp hữu ích khác như lớp JSON để xử lý kiểu dữ liệu JSON, lớp Function để làm việc với hàm, lớp Error để xử lý lỗi, ...

5.6 MẢNG

5.6.1 Mảng được sử dụng để lưu trữ nhiều giá trị vào một biến

Khai bảo mảng sử dụng **dấu ngoặc vuông []** với các phần tử, hoặc sử dụng **new Array**:

```
var thanhpho = ["Hà Nội", "Thái Bình", "Hưng Yên"];
var nghenghiep = new Array("Giáo viên", "Bác sỹ", "Doanh nhân");
```

Để truy cập đến từng phần tử của mảng, sử dụng dấu ngoặc vuông [] với phần tử của mảng. Các phần tử của mảng có thể có các kiểu khác nhau, và cũng có thể la là một đối tượng, một mảng khác hoặc một hàm:

```
thanhpho[3] = "Quảng Ninh"
thanhpho[4] = 29;
thanhpho[5] = () => "Vĩnh Phúc"
```

Duyệt các phần tử của mảng bằng cách dùng vòng lặp for. Chú ý phần tử đầu tiên của mảng bắt đầu từ 0:

```
for (i = 0; i < thanhpho.length; i++) {
    console.log(thanhpho[i])
}
```

Kết quả:

```
Hà Nội
Thái Bình
Hưng Yên
Quảng Ninh
29
[Function (anonymous)]
```

Ngoài ra cũng có thể sử dụng phương thức forEach() của mảng. Ví dụ:

```
nghenghiep.forEach((item)=>console.log(item));
```

Kết quả:

```
Giáo viên
Bác sỹ
Doanh nhân
```

Mảng là một dạng đặc biệt của đối tượng, và từ khóa **typeof** cũng sẽ trả về kiểu là "**object**".

5.6.2 Phương thức

Mảng cũng là một đối tượng vì vậy có rất nhiều phương thức có sẵn. Ví dụ phương thức **toString()** để chuyển mảng về dạng chuỗi. Phương thức **join()** cũng chuyển một mảng thành chuỗi nhưng có thể khai báo thêm từ phân cách các phần tử:

```
var thanhpho = ["Hà Nội", "Thái Bình", "Hưng Yên"];
console.log(thanhpho.toString()) //Hà Nội,Thái Bình,Hưng Yên
console.log(thanhpho.join("-")) // Hà Nội-Thái Bình-Hưng Yên
```

Phương thức **push()** để thêm phần tử vào cuối mảng, phương thức pop() sẽ xóa phần tử cuối cùng của mảng, và trả về phần tử đó của mảng**:**

```
var thanhpho = ["Hà Nội", "Thái Bình", "Hưng Yên"];
console.log(thanhpho) //[ 'Hà Nội', 'Thái Bình', 'Hưng Yên' ]

thanhpho.push("Quảnh Ninh") //Thêm phần tử vào cuối mảng
console.log(thanhpho) //[ 'Hà Nội', 'Thái Bình', 'Hưng Yên', 'Quảnh Ninh' ]

thanhpho.pop() //Xóa phần tử cuối của mảng
console.log(thanhpho) //[ 'Hà Nội', 'Thái Bình', 'Hưng Yên' ]

thanhpho.shift() //Xóa phần tử đầu tiên của mảng
console.log(thanhpho) //[ 'Thái Bình', 'Hưng Yên' ]

thanhpho.unshift("Tuyên Quang") //Thêm phần tử vào đầu của mảng
console.log(thanhpho) //[ 'Tuyên Quang', 'Thái Bình', 'Hưng Yên' ]
```

Lưu ý, Để xóa phần tử của mảng có thể sử dụng lệnh **delete**. Tuy nhiên, sử dụng **delete** đôi khi sẽ tạo ra các **hố** (hole) trong mảng là các phần không gian trong mảng không lưu trữ dữ liệu gì.

Trường hợp muốn xoá nên sử dụng shift() hoặc pop(). Các phương thức này xóa phần tử của mảng và cũng trả về giá trị vừa được xóa. Ngoài ra mảng còn được hỗ trợ rất nhiều các phương thức hữu ích khác như **splice()** để thêm hoặc thay thế phần tử vào vị trí bất kỳ trong mảng; phương thức **concat()** để ghép 1 mảng vào một mảng khác; phương thức **slice()** dùng để cắt một phần của mảng sang một mảng mới...

5.6.3 Sắp xếp mảng

Phương thức sort() để sắp xếp các phần tử trong mảng:

```
var thanhpho = ["Hà Nội", "Thái Bình", "Hưng Yên"];
thanhpho.sort()
console.log(thanhpho) //[ 'Hà Nội', 'Hưng Yên', 'Thái Bình' ]
```

Các phần tử của mảng có thể so sánh được với nhau thì hàm sort mới có tác dụng, trường hợp các phần tử của mảng là các đối tượng, hệ thống sẽ không biết cách để so sánh đối tượng, khi đó cần cung cấp một hàm so sánh khi sắp xếp. Ví dụ cho một mảng các đối tượng sinh viên:

Sắp xếp mảng đối tượng:

```
var dsSinhvien = [
    {ten:"Anh", tuoi:20},
    {ten:"Phuong", tuoi:21},
    {ten:"Minh", tuoi:19},
    {ten:"Binh", tuoi:20}
];
dsSinhvien.sort(function(a, b){return a.tuoi - b.tuoi});
console.log(dsSinhvien)
```

Để sắp xếp theo tuổi của sinh viên, hàm so sánh được viết như sau:

```
dsSinhvien.sort(function(a, b){return a.tuoi - b.tuoi});
```

Kết quả sau khi sắp xếp:

```
[
  { ten: 'Minh', tuoi: 19 },
  { ten: 'Anh', tuoi: 20 },
  { ten: 'Binh', tuoi: 20 },
  { ten: 'Phuong', tuoi: 21 }
]
```

Nếu muốn sắp xếp theo chiều ngược lại, có thể sửa hàm so sánh. Một cách khác là dùng phương thức **reverse()** của mảng để đảo các phần tử ngược lại so với thứ tự sau khi sắp xếp.

5.6.4 Duyệt mảng

Ngoài việc duyệt mảng bằng lệnh for hoặc phương thức forEach đã trình bày ở trên. JavaScript còn cho phép duyệt qua từng phần tử của mảng và thực hiện một công việc khác đồng thời trong khi duyệt. Ví dụ phương thức **map()** duyệt mảng và tạo ra một mảng mới:

```
var thanhpho = ["Hà Nội", "Thái Bình", "Hưng Yên"];
var thanhpho1 = thanhpho.map((value, index) => index + 1 + ". " + value);
console.log(thanhpho1) // [ '1. Hà Nội', '2. Thái Bình', '3. Hưng Yên' ]
```

Phương thức **filter()** cũng duyệt và tạo ra mảng mới nhưng lựa chọn những phần tử phù hợp, ví dụ chỉ chọn các phần tử mà có chữ "i":

```
var thanhpho2 = thanhpho.filter((value, index) => value.includes("i"));
console.log(thanhpho2) // [ 'Hà Nội', 'Thái Bình' ] có chữ i
```

Phương thức **reduce()** cũng dùng để duyệt và chuyển mảng thành một phần tử duy nhất:

```
var thanhpho3 = thanhpho.reduce((total, value, index) => total + "|" + value);
console.log(thanhpho3) // Hà Nội|Thái Bình|Hưng Yên
```

Phương thức **every()** để kiểm tra toàn bộ các phần tử của mảng có thoả mãn một điều kiện gì đó hay không:

```
var thanhpho4 = thanhpho.every((value) => value.length >= 6);
console.log(thanhpho4) //true vì mọi phần tử đều có độ dài hơn hoặc bằng 6
```

5.7 CÂU ĐIỀU KIỆN

Trong JavaScript, có các **hai loại câu điều kiện**:

- Câu điều kiện **if/else** để chỉ định một khối mã sẽ được thực thi, nếu một điều kiện được chỉ định là đúng/sai.
- Câu lệnh **switch** để chỉ định nhiều khối mã thay thế sẽ được thực thi.

5.7.1 Câu lệnh if else

Sử dụng **if** để chỉ định một khối mã JavaScript sẽ được thực thi nếu một **điều kiện đúng**. Sử dụng **else** để chỉ định một khối mã sẽ được thực thi nếu **điều kiện sai**. Cú pháp:

```
if (điều kiện) {
 //Khối lệnh sẽ thực hiện khi điều kiện đúng
}
else {
 //Khối lệnh sẽ thực hiện khi điều kiện sai
}
```

Ví dụ thực hiện lời nhắn *"Bạn đủ điều kiện bầu cử"* nếu tuổi lớn hơn 18

```
if (tuoi >= 18) {
    console.log("Bạn đủ điều kiện bầu cử")
} else {
    console.log("Bạn CHƯA đủ điều kiện bầu cử")
}
```

Sử dụng **else if** để chỉ định một điều kiện mới nếu điều kiện đầu tiên là sai. Cú pháp:

```
if (điều kiện 1) {
 //Khối lệnh sẽ thực hiện khi điều kiện 1 đúng
}
else if (điều kiện 2) {
 //Khối lệnh sẽ thực hiện khi điều kiện 2 đúng
}
```

```
else {
 //Khối lệnh sẽ thực hiện khi 2 điều kiện trên là sai
}
```

Ví dụ nếu trước 11 giờ trưa, hãy đưa ra một lời *"Chào buổi sáng"*, nếu sau 19 giờ thì chào *"Chào buổi tối"*, cá thời điểm khác trong ngày thì đưa ra lời chúc *"Chúc một ngày tốt lành"*. Mã lệnh sẽ viết như sau:

```
const d = new Date();
let hour = d.getHours(); //Hour sẽ nhận giá trị từ 0 đến 23
if (hour < 11) {
    console.log("Chào buổi sáng")
} else if (hour >19) {
    console.log("Chào buổi tối")
} else {
    console.log("Chúc một ngày tốt lành")
}
```

5.7.2 Câu lệnh switch

Lệnh **switch** được sử dụng để thực hiện các hành động khác nhau dựa trên các điều kiện khác nhau. Cú pháp của câu lệnh **switch**:

```
switch(Biểu thức) {
 case x:
 // Khối lệnh
 break;
 case y:
 // Khối lệnh
 break;
 default:
 // Khối lệnh
}
```

Tùy thuộc vào giá trị biểu thức với giá trị được khai báo trong **case** nào mà khối lệnh tương ứng sẽ được thực hiện. Ví dụ sử dụng số ngày trong tuần để tính tên ngày trong tuần:

```
switch (new Date().getDay()) {
 case 0:
   day = "Chủ nhật";
   break;
 case 1:
   day = "Thứ 2";
   break;
 case 2:
   day = "Thứ 3";
   break;
 case 3:
   day = "Thứ 4";
   break;
 case 4:
   day = "Thứ 5";
   break;
 case 5:
```

```
    day = "Thứ 6";
    break;
  case 6:
    day = "Thứ 7";
    break;
default:
    console.log("Không phải ngày trong tuần")
}
```

Khi JavaScript gặp lệnh **break** , nó sẽ thoát ra khỏi khối lệnh **switch**. Lưu ý, nếu bỏ qua câu lệnh **break**, trường hợp tiếp theo sẽ được thực thi ngay cả khi đánh giá không khớp với trường hợp. Từ khóa **default** chỉ định mã để chạy nếu không có trường hợp nào phù hợp phía trên. Tuy nhiên **default** không hẳn cần khai báo cuối cùng.

Đôi khi sẽ muốn các trường hợp cùng thực hiện một mã lệnh có thể gom bằng cách sử dụng các **case** liên tiếp nhau. Trong ví dụ dưới đây, **case 4 và 5** chia sẻ cùng một khối mã và **case 0 và 6** chia sẻ một khối mã khác:

```
switch (new Date().getDay()) {
    case 0:
    case 6:  //Các case có thể viết trên các dòng
        text = "Ngày cuối tuần, nghỉ ngơi thôi";
        break;
    case 1,2,3,4,5:  //Các case có thể được gom trên một dòng
        text = "Chúc bạn một ngày làm việc hiệu quả";
        break;
}
```

5.8 VÒNG LẶP

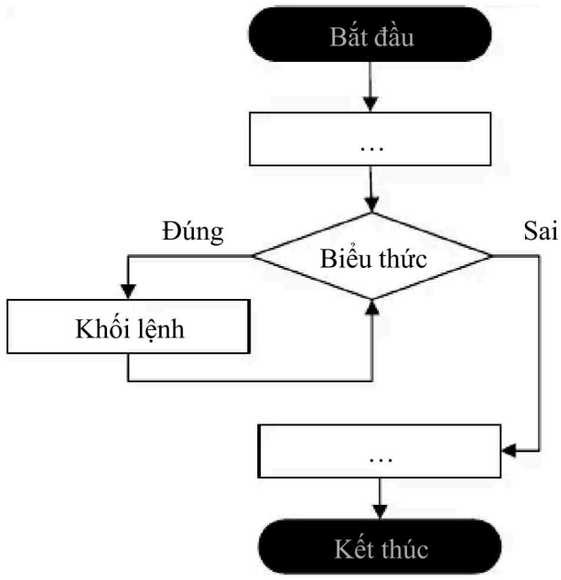
Vòng lặp là một cú pháp lập trình cho phép thực thi một khối mã nhiều lần. JavaScript hỗ trợ các loại vòng lặp khác nhau như: **while**, **do/while**, **for**, **for/in**, và **for/of**.

5.8.1 Vòng lặp while

Vòng lặp **while** giúp thực hiện một **khối lệnh** miễn là một **biểu thức** được chỉ định là đúng. Cú pháp của vòng lặp **while** như sau:

```
while (biểu thức) {
   khối lệnh
}
```

Sơ đồ khối của vòng lặp **while**:



Ví dụ vòng lặp sẽ chạy, lặp đi lặp lại, miễn là một biến (i) nhỏ hơn 10:

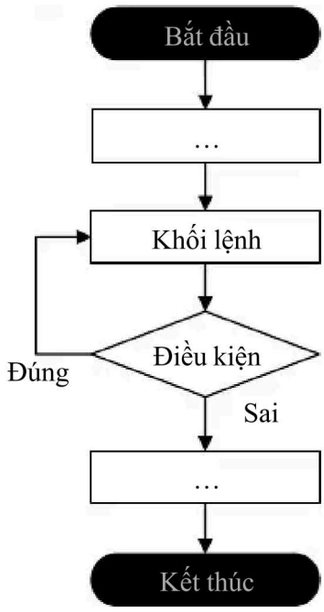
```
var menu = ["Trang chủ", "Bài viết", "Dự án", "Đăng ký khóa học"];
let i = 0;
while (i < menu.length) {
    console.log(menu[i])
    i++;
}
```

5.8.2 Vòng lặp do while

Vòng lặp **do/while** hoạt động tương tự vòng lặp **while**. Vòng lặp này sẽ thực thi **khối lệnh** một lần, trước khi kiểm tra **biểu thức** đúng không, sau đó nó sẽ lặp lại vòng lặp miễn là **biểu thức** đúng:

```
do {
   khối lệnh
} while (biểu thức);
```

Sơ đồ khối của vòng lặp **do/while**:



Ví dụ dưới đây sử dụng một vòng lặp **do/while**. Vòng lặp sẽ luôn được thực thi ít nhất một lần, ngay cả khi biểu thức sai, bởi vì **khối lệnh** được thực thi trước khi **biểu thức** được kiểm tra. Ví dụ:

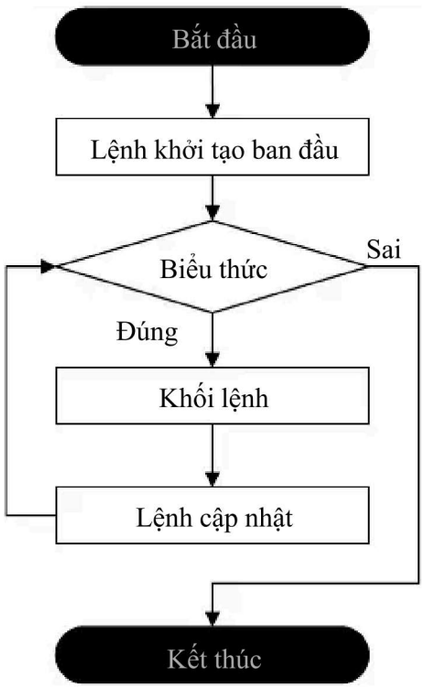
```
var menu = ["Trang chủ", "Bài viết", "Dự án", "Đăng ký khóa học"];
let i = 0;
do {
    console.log(menu[i])
    i++;
} while (i < menu.length)
```

- **Vòng lặp for**

Vòng lặp **for** có cú pháp như sau:

```
for (lệnh khởi tạo đầu; biểu thức; lệnh cập nhật) {
   khối lệnh
}
```

Sơ đồ khối của vòng lặp **for**:



Trong cấu trúc của vòng lặp **for**, **lệnh khởi tạo** được thực thi một lần sau đó **biểu thức** sẽ được kiểm tra, nếu **biểu thức** là đúng thì **khối lệnh** được thực hiện, tiếp theo **lệnh cập nhật** được thực thi trước khi **biểu thức** được kiểm tra lần tiếp theo để xác định xem **khối lệnh** có được tiếp tục thực hiện hay không. Ví dụ:

```
var menu = ["Trang chủ", "Bài viết", "Dự án", "Đăng ký khóa học"];
for (let i = 0; i < menu.length; i++) {
    console.log(menu[i])
}
```

Vòng lặp **for/in** lặp qua các thuộc tính của một đối tượng. Ví dụ:

```
var sinhvien = {
    ho: "Pham",
    dem: "Duong",
    ten: "Minh",
    tuoi: 21,
    quequan: "Hai Duong",
    lop: "Cong nghe thong tin 62"
};
for (x in sinhvien) {
    console.log(sinhvien[x])
}
```

Vòng lặp **for/of** cho phép lặp qua các cấu trúc dữ liệu có thể lặp lại như Mảng, Chuỗi, Bản đồ, Danh sách, v.v. Ví dụ khi duyệt mảng:

```
var menu = ["Trang chủ", "Bài viết", "Dự án", "Đăng ký khóa học"];
for (let item of menu) {
    console.log(item)
}
```

Vòng lặp **for/of** cũng được sử dụng để duyệt các ký tự trong chuỗi:

```
var menu = "Thiết kế Web";
for (let item of menu) {
    console.log(item)
}
```

5.8.3 Câu lệnh break và continue

Các câu **break** và **continue** này cũng được sử dụng để điều khiển vòng lặp theo ý muốn mà không cần hoàn toàn phụ thuộc vào các điều kiện kiểm tra.

Câu lệnh **break** có thể được sử dụng để kết thúc vòng lặp. Ví dụ vòng lặp này sẽ dừng khi gặp ký tự W:

```
var menu = "Thiết kế Web";
for (let item of menu) {
    if (item === "W") break;
    console.log(item)
}
```

Câu lệnh **continue** dùng để bỏ qua một lượt lặp. Ví dụ việc lặp chuỗi này sẽ bỏ qua toàn bộ các ký tự **dấu cách (space):**

```
var menu = "Thiết kế Web";
for (let item of menu) {
    if (item === " ") continue;
    console.log(item)
}
```

Các **break** và **continue** là những lệnh JavaScript có thể "nhảy ra" một khối mã để đến một câu lệnh được gán nhãn. Nếu sử dụng câu lệnh **break** bên trong một vòng lặp lồng nhau, nó sẽ chỉ thoát ra khỏi vòng lặp đó, không thoát ra khỏi vòng lặp có chứa vòng lặp ngoài. Để t**hoát ra khỏi vòng lặp lồng nhau** từ bên trong, cần sử dụng **break kèm theo nhãn**.

5.9 XỬ LÝ LỖI

5.9.1 try catch

Trong quá trình thực hiện mã lệnh, nếu một lỗi xuất hiện, chương trình sẽ dừng việc thực thi, các câu lệnh phía dưới sẽ không được thực hiện. Ví dụ:

```
console.log(x)
console.log("Kết thúc")
```

Ví dụ với mã lệnh trên, biến x chưa được khai báo, vì vậy chương trình sẽ phát sinh lỗi:

```
main.js:23
    console.log(x)
                ^
ReferenceError: x is not defined
```

Hệ thống sẽ báo một lỗi tham chiếu: x được sử dụng nhưng chưa được định nghĩa. Lối xảy ra khiến câu lệnh phía dưới sẽ không được thực hiện. Để đảm bảo những lỗi phát sinh trong chương trình được xử lý, cần sử dụng cấu trúc **try..catch.** Cấu trúc xử lý lỗi bao gồm 2 lệnh,

trong đó lệnh **try** cho phép xác định một khối mã được kiểm tra lỗi trong khi nó đang được thực thi. Câu lệnh **catch** cho phép khai báo khối lệnh sẽ được thực thiện nếu khối lệnh thực thi trong **try** phát sinh lỗi. Cú pháp cụ thể như sau:

```
try {
 khối lệnh thực thi
}
catch(err) {
 khối lệnh sẽ được thực hiện nếu khối lệnh thực thi phát sinh lỗi
}
```

Ví dụ:

```
try {
    console.log(x)
} catch (err) {
    console.log(err.name + ": " + err.message)
}
console.log("Kết thúc")
```

Kết quả khi thực thi:

```
ReferenceError: x is not defined
End
```

Khi xảy ra lỗi, JavaScript tạo một đối tượng Error với hai thuộc tính: **tên (name)** và **thông báo (message)**. Các thông tin này giúp nhà phát triển biết được lỗi đã phát sinh. Với việc sử dụng dụng **try...catch**, các câu lệnh phía dưới vẫn được thực hiện.

5.9.2 Ném lỗi

Câu lệnh **throw** cho phép **tạo một lỗi**. Nói chung, lỗi thường tự phát sinh trong quá trình hoạt động, tuy nhiên người lập trình có thể chủ động ném ra các lỗi để điều khiển luồng chương trình theo ý muốn. Một lỗi ném ra có thể là một chuỗi, số, hoặc một đối tượng.

```
throw "Không hợp lệ"; // Ném một chuỗi
throw 500;            // Ném một số
throw new Date()      // Ném một đối tượng
```

Nếu sử dụng **throw** cùng với **try** và **catch**, có thể kiểm soát luồng chương trình và tạo thông báo lỗi tùy chỉnh.

5.9.3 Lệnh finally

Câu lệnh **finally** cho phép tạo ra cấu trúc xử lý lỗi **try**...**catch**...**finally**. Các câu lệnh trong khối **finally** sẽ được đảm bảo luôn thực hiện trong mọi trường hợp.

```
try {
 khối lệnh thực thi
}
catch(err) {
 khối lệnh sẽ được thực hiện nếu khối lệnh thực thi phát sinh lỗi
}
finally {
```

```
 Khối lệnh sẽ luôn được thực hiện
}
```

Để hiểu rõ hơn về finally, xét ví dụ sau:

```
try {
    console.log(x)
} catch (err) {
    throw 4;
}
console.log("Kết thúc")
```

Trong ví dụ trên, chữ **"Kết thúc"** sẽ không được hiển thị ra vì có lỗi xảy ra trong khối **catch** (do lệnh **throw 4** tạo ra). Để đảm bảo cho lệnh **console.log("Kết thúc")** được thực thi, sử dụng **finally** như sau:

```
try {
    console.log(x)
} catch (err) {
    throw 4;
}
console.log("Kết thúc")
```

5.10 ĐỐI TƯỢNG

Các cách để tạo đối tượng:

- Sử dụng {}
- Sử dụng Constructor
- Sử dụng Object.create()
- Sử dụng new Object
- Sử dụng new + tên lớp

5.10.1 Tạo đối tượng bằng {}

```
let me = {ten: "Phạm Dương Minh", email: "phamduongminh@mail.com", diachi: "207
Giải Phóng", tuoi: 22};
```

Việc tạo đối tượng ban đầu có thể khai báo rỗng, sau đó khai báo các thuộc tính và phương thức:

```
let me = {};
me.ten = "Phạm Dương Minh";
me.email = "phamduongminh@mail.com";
me.diachi = "207 Giải Phóng";
me.tuoi = 22;
```

Các thuộc tính có thể được truy cập bằng toán tử **chấm "."** hoặc dấu **ngoặc vuông**:

```
tên-đối-tượng.tên-thuộc tính
tên-đối-tượng[tên-thuộc tính]
```

Với **cách thứ 2**, có thể để một biểu thức tính toán vào trong **ngoặc vuông**, điều mà cách thứ nhất không thể. Ví dụ để duyệt qua các thuộc tính, dùng vòng lặp **for/in,** cú pháp của **for/in** như sau:

```
for (let tên-biến in đối-tượng) {
   Khối lệnh
}
```

Khi sử dụng vòng lặp **for/in** như trên, mỗi lần lặp sẽ trả về **tên-biến** chứa tên các thuộc tính của **đối-tượng,** khi đó cần sử dụng cách thứ 2 để truy cập vào giá trị:

```
let me = {ten: "Phạm Dương Minh", email: "phamduongminh@mail.com", diachi: "207
Giải Phóng", tuoi: 22};
for (let i in me) {
    console.log(me[i])
}
```

Để **thêm thuộc tính** chỉ cần thực hiện phép **gán giá trị** vào thuộc tính mới:

```
me.kynang = "JavaScript, Java, NodeJS, Web"
```

Việc **thêm phương thức** cũng được thực hiện tương tự, chú ý trong các phương thức muốn truy cập vào các thuộc tính của chính đối tượng, sử dụng từ khóa **this**:

```
me.hienthongtin = function () {
    console.log(this.ten + " : " + this.email)
}
myInfo.hienthongtin()
```

Để xóa thuộc tính hoặc phương thức của một đối tượng, sử dụng lệnh **delete**:

```
delete me.hienthongtin;
me.hienthongtin()  //Phương thức này sẽ không còn hiệu lực
```

5.10.2 Tạo đối tượng bằng phương thức khởi tạo

Tạo đối tượng bằng phương thức khởi tạo:

```
function User(ten, email, diachi, tuoi) {
    this.ten = ten;
    this.email = email;
    this.diachi = diachi;
    this.tuoi = tuoi;
    this.hienthongtin = function () {
        console.log(this.ten + " : " + this.email)
    }
}
```

Từ phương thức tạo này tạo ra lớp **User**, từ đó có thể tạo ra các đối tượng khác nhau, ví dụ đối tượng **me**:

```
let me = new User("Phạm Dương Minh", "phamduongminh@mail.com", "207 Giải Phóng",
22);
```

Để thêm một **thuộc tính**, hoặc **phương thức** cho đối tượng **me**, chỉ cần gán giá trị, hàm cho thuộc tính đó, ví dụ:

```
me.dienthoai = "02939392121"
me.kiemtra = function () {
    if (this.tuoi > 18)
```

```
        return true;
    else
        return false
};
```

Tuy nhiên với việc thêm như trên chỉ có đối tượng me có giá trị của thuộc tính mới, trong trường hợp muốn thêm cho cả lớp User, sử dụng prototype, ví dụ thêm thuộc tính **quoctich** và phương thức **kiemtra()**:

```
User.prototype.quoctich = "Việt Nam";
User.prototype.kiemtra = function () {
    if (this.tuoi > 18)
        return true;
    else
        return false
};
```

Khi đó toàn bộ các đối tượng được tạo ra bởi lớp **User** sẽ đều có thuộc tính **quoctich** và phương thức **kiemtra()**.

5.10.3 Tạo đối tượng bằng new [class Name]

Tạo đối tượng trong Javascript bằng cách định nghĩa lớp (class) sử dụng từ khóa **class**, sau đó sử dụng lớp đó để tạo ra các đối tượng. Cú pháp khai báo lớp như sau:

```
class Tên-lớp {
   constructor() { ... }
}
```

Ví dụ tạo ra lớp User:

```
class User {
    constructor(ten, email, diachi, tuoi) {
        this.ten = ten;
        this.email = email;
        this.diachi = diachi;
        this.tuoi = tuoi;
    }
    hienthongtin() {
        console.log(this.ten + " : " + this.email)
    }
}
```

Chú ý **class** luôn phải khai báo khi tạo đối tượng trước do class không được **hoist** giống như khai báo biến. Sau khi lớp đã có, có thể sử dụng để tạo đối tượng, ví dụ đối tượng me:

```
let me = new User("Phạm Dương Minh", "phamduongminh@mail.com", "207 Giải Phóng",
22);
```

5.10.4 Tính chất kế thừa

Khi tạo đối tượng bằng cách sử dụng {}, việc kế thừa sẽ được triển khai bằng cách sử dụng **__proto__**. Cách viết như sau:

```
let me = {ten: "Phạm Dương Minh", email: "phamduongminh@mail.com", diachi: "207
Giải Phóng", tuoi: 22};
let sinhvien = {masv: 1578957}
sinhvien.__proto__ = me;
```

Khi tạo đối tượng bằng **phương thức khởi tạo**, nếu muốn triển khai tính kế thừa, có thể sử dụng hàm **call()**. Ví dụ tạo ra lớp Student kế thừa từ lớp User:

```
function User(ten, email, diachi, tuoi) {
    this.ten = ten;
    this.email = email;
    this.diachi = diachi;
    this.tuoi = tuoi;
    this.hienthongtin = function () {
        console.log(this.ten + " : " + this.email)
    }
}

function Student(ten, email, diachi, tuoi, masv) {
    User.call(this, ten, email, diachi, tuoi)  //Kế thừa
    this.masv = masv;
}

const sinhvien = new Student("Phạm Dương Minh", "phamduongminh@mail.com", "207
Giải Phóng", 22, 1578957);
sinhvien.hienthongtin() //Sử dụng phương thức của lớp User
```

Khi tạo đối tượng bằng khai báo class, việc triển khai tính kế thừa được thực hiện bằng cách sử dụng từ khóa **extends**:

```
class Student extends User {
    constructor(ten, email, diachi, tuoi, masv) {
        super(ten, email, diachi, tuoi)
        this.masv = masv;
    }
    hienthongtin() {
        super.hienthongtin();
        console.log(this.masv)
    }
}
```

Trong lớp con, nếu muốn gọi phương thức của lớp cha thì cần sử dụng từ khóa **super.** Lớp con cũng có thể **khai báo lại** một phương thức, hay thuộc tính đã có của lớp cha. Ví dụ trên thì lớp Student khai báo lại phương thức **hienthongtin()** đồng thời gọi phương thức này bằng cách sử dụng **super**.

5.10.5 Phương thức tĩnh

Phương thức tĩnh là phương thức được khai báo với từ khóa **static**:

```
class Car {
   constructor(name) {
       this.name = name;
   }
   static hello() {
```

```
        return "Hello!!";
    }
}
```

Phương thức tĩnh không thể được gọi thông qua đối tượng mà phải gọi thông qua class:

```
let myCar = new Car("Ford");
console.log(Car.hello()); //OK
console.log(myCar.hello()); //error
```

5.10.6 Phương thức getter và setter

Phương thức getter tương tự như các phương thức khác, tuy nhiên sử dụng tương tự như cú pháp truy cập vào thuộc tính. Phương thức getter được khai báo bằng từ khóa **get:**

```
const person = {
    firstName: "John",
    lastName: "Doe",
    language: "en",
    get lang() {
        return this.language;
    }
};
console.log(person.lang);
```

Phương thức setter được khai báo bằng từ khóa **set**. Khác với các phương thức khác, setter được gọi thông qua phép gán. Như trong ví dụ trên phương thức setter lang(language) sẽ được gọi khi gán person.lang = "en".

```
const person = {
    firstName: "John",
    lastName: "Doe",
    language: "",
    set lang(language) {
        this.language = language;
    }
};
person.lang = "en";
console.log(person.language);
```

CÂU HỎI ÔN TẬP LÝ THUYẾT

1. Làm cách nào để tạo một hàm trong JavaScript?

A. function:myFunction()

B. function = myFunction()

C. function myFunction()

D. function:=myFunction()

2. Làm thế nào để gọi một hàm có tên là "myFunction"?

A. call function myFunction()
B. myFunction()
C. call myFunction()
D. function.myFunction()

3. Làm thế nào để viết một câu lệnh IF trong JavaScript?
A. if (i == 5)
B. if i == 5 then
C. if i = 5 then
D. if i = 5

4. Làm thế nào để viết một câu lệnh IF để thực thi một số mã nếu "i" KHÔNG bằng 5?
A. if i =! 5 then
B. if i <> 5
C. if (i != 5)
D. if (i <> 5)

5. Câu lệnh nào được sử dụng để khai báo biến trong JavaScript?
A. Let
B. Var
C. Const
D. tất cả đều đúng

6. Kết quả của lệnh console.log(2 + "2") sẽ là gì?
A. 4
B. "22"
C. NaN
D. "2 + 2"

7. Sử dụng toán tử nào để so sánh giá trị và kiểu dữ liệu của hai biến trong JavaScript?
A. ==
B. ===

C. !=

D. <>

8. Lệnh nào được sử dụng để lấy độ dài của một chuỗi trong JavaScript?

A. length()

B. size()

C. count()

D. lấy giá trị từ phần tử cuối cùng

9. Sử dụng lệnh nào để tạo một mảng trong JavaScript?

A. array()

B. list()

C. []

D. set()

10. Sử dụng lệnh nào để truy xuất một phần tử trong một mảng trong JavaScript?

A. get()

B. set()

C. ind()

D. index()

TÀI LIỆU THAM KHẢO

[1] Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics (2018), Jennifer Robbins, O'Reilly Media

[2] JavaScript: The Definitive Guide: Master the World's Most-Used Programming Language (2020), David Flanagan, O'Reilly Media

[3] Professional JavaScript® for Web Developers (2020), Matt Frisbie, Wrox

CHƯƠNG 6: JAVASCRIPT TRONG THIẾT KẾ WEBSITE

Chương này sẽ giới thiệu về đối tượng tài liệu (DOM) nhằm thực hiện việc ghi nhận, thay đổi, thêm hoặc xóa các phần tử trong HTML. Người học sẽ được tìm hiểu cách sử dụng JavaScript để xử lý các sự kiện, xác thực các biểu mẫu trong một trang Web. Bên cạnh đó, các kiểu dữ liệu (JSON, XML) & cách thức tải dữ liệu từ các hệ thống Website khác không cần tải lại trang (AJAX) cũng sẽ được đề cập.

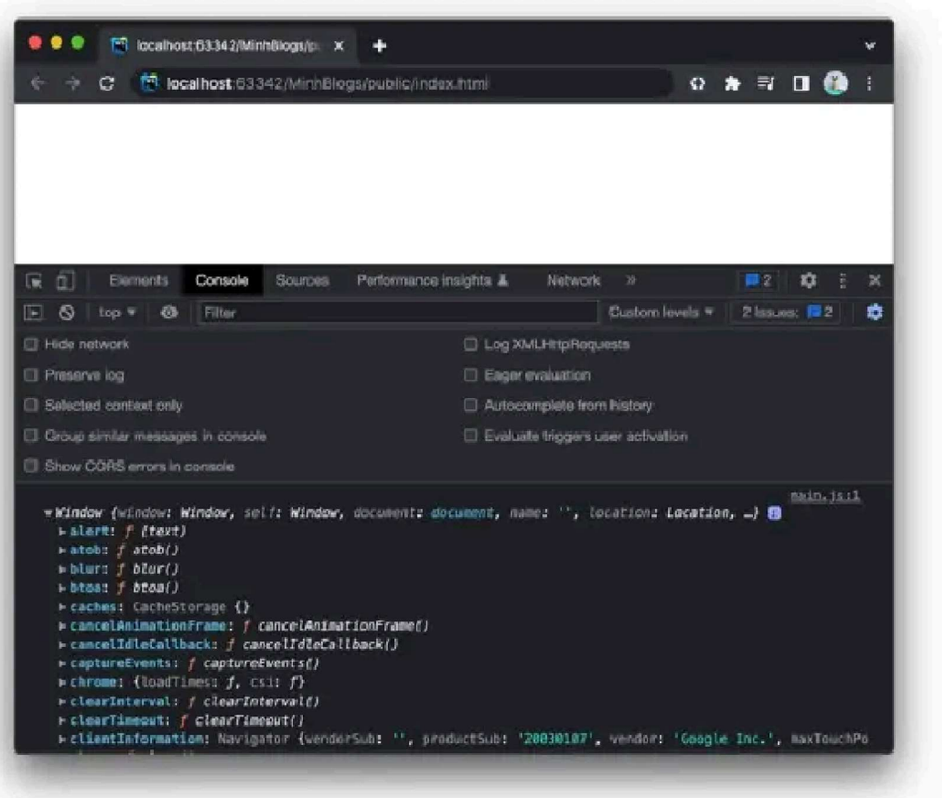
6.1 ĐỐI TƯỢNG CỬA SỔ

6.1.1 Đối tượng cửa sổ là gì?

Khi lập trình Javascript, **cửa sổ (window)** là một đối tượng có thể truy cập bất kỳ lúc nào. Đối tượng **window** cung cấp các đối tượng và phương thức để lập trình trong thiết kế Website. In thử thông tin về đối tượng **window** bằng lệnh sau:

```
console.log(window)
```

Có thể thấy thông tin về đối tượng window được in ra như sau:



Hình 6-1 Thông tin về đối tượng window

Có thể nhận thấy đối tượng **window** có rất nhiều phương thức hữu ích thông dụng. Ví dụ như:

- **window.alert()**: Hiện thị một thông báo.

- **window.open():** Mở một cửa sổ mới.
- **window.close()**: Đóng cửa sổ hiện tại.
- **window.moveTo()**: Di chuyển tới cửa sổ.
- **window.resizeTo()**: Thay đổi kích thước cửa sổ.
- ..

Bên cạnh đó đối tượng **window** có chứa rất nhiều thuộc tính là các đối tượng. Ví dụ như

- **Caches**: Đối tượng chứa thông tin về bộ nhớ đệm.
- **Document**: Đối tượng chứa thông tin về tài liệu HTML.
- **History**: Đối tượng chứa thông tin về lịch sử.
- **Location**: Đối tượng chứa thông tin về địa chỉ Web.
- **Navigator**: Đối tượng trình duyệt.
- **Screen**: Đối tượng chứa thông tin về màn hình.
- ...

Điều đặc biệt là để truy cập vào thuộc tính và sử dụng các phương thức của window thì không cần phải ghi cụ thể **window.tên-thuộc-tính** hoặc **window.tên-phương-thức()** mà chỉ cần gọi trực tiếp **tên-thuộc-tính** hoặc **tên-phương-thức()**. Ví dụ sử dụng phương thức alert() để in ra kích thước của trình duyệt, thay vì viết mã:

```
window.alert(window.outerWidth + ":" + window.outerHeight)
```

chỉ cần viết:

```
window.alert(outerWidth + ":" + outerHeight)
```

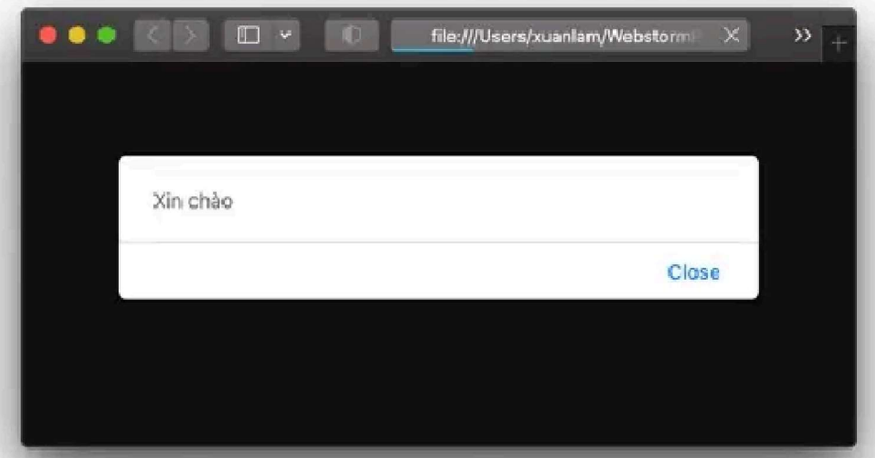
Trong lệnh trên ta đã sử dụng phương thức **alert()** của đối tượng **window** để in ra thuộc tính **outerWidth** và **outerHeight** của đối tượng **window**. Đó chính là kích thước chiều rộng và chiều cao của cửa sổ trình duyệt. Đối tượng window còn có nhiều các phương thưc và thuộc tính quan trọng khác sẽ tìm hiểu trong phần dưới đây.

6.1.2 Các phương thức quan trọng

Đối tượng **window** cung cấp rất nhiều các phương thức quan trọng. Như trong phần trên có giới thiệu phương thức **alert()** cho phép hiển thị một **hộp thoại thông báo** (hoặc cảnh báo) nhằm gửi thông tin đến cho người dùng. Hộp thoại như một công cụ giúp Website có thể **"thoại"** hay nói cách khác là giao tiếp với người dùng. Ví dụ thông báo hiển thị một lời chào tới người dùng:

```
alert("Xin chào")
```

Khi **hộp thoại thông báo** bật lên, người dùng sẽ bấm **OK** hoặc **Close** (tùy trình duyệt) để tiếp tục. Hộp thông báo sẽ hiển thị khác nhau trên các trình duyệt khác nhau, ví dụ trên trình duyệt Safari, thông báo sẽ xuất hiện như sau:



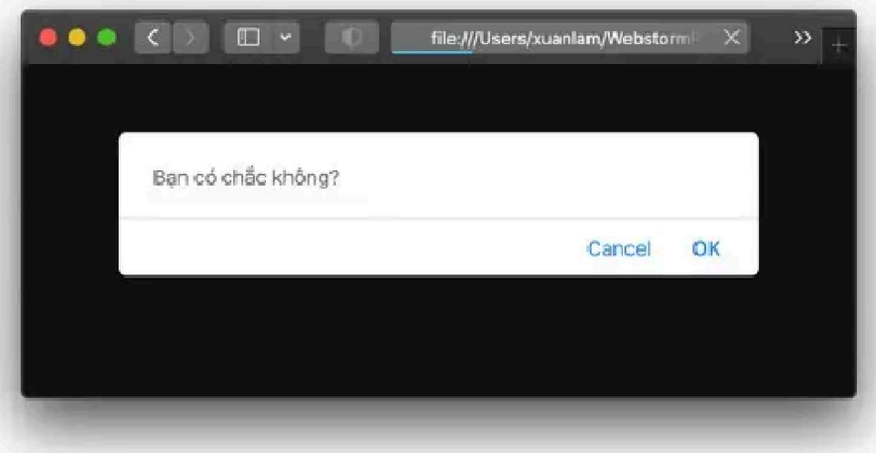
Hình 6-2 Hộp thoại thông báo

Để hiển thị ngắt dòng bên trong hộp bật lên, hãy sử dụng dấu gạch chéo (\) theo sau là ký tự n. Ví dụ:

```
alert("Xin chào\n Chúc một ngày tốt lành")
```

Phương thức **confirm()** giúp hiển thị ra một **hộp thoại xác nhận** để người dùng xác minh hoặc chấp nhận một cái gì đó.

```
let luachon = confirm("Bạn có chắc không");
```

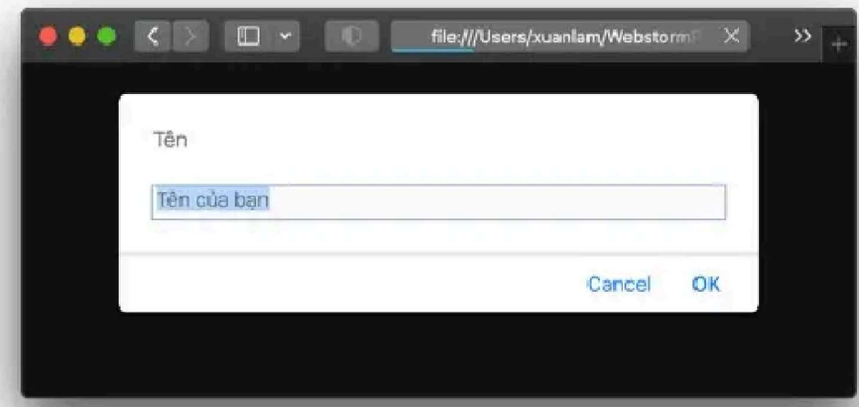


Hình 6-3 Hộp thoại xác nhận

Khi **hộp thoại xác nhận** bật lên, người dùng sẽ phải nhấp vào "OK" hoặc "Cancel" để tiếp tục. Nếu người dùng nhấp vào "OK", phương thức confirm() sẽ trả về **true**. Nếu người dùng nhấp vào **"Hủy"**, phương thức **confirm**() sẽ trả về **false.** Như vậy có thể biết được người dùng chọn phương án nào.

Phương thức **prompt()** giúp hiển thị **hộp thoại nhập** để lấy giá trị từ người dùng nhập. Khi **hộp thoại nhập** bật lên, người dùng sẽ phải nhấp vào "OK" hoặc "Cancel" để tiếp tục sau khi nhập giá trị đầu vào. Nếu người dùng nhấp vào "OK", hộp thoại sẽ trả về giá trị đầu vào. Nếu người dùng nhấp vào "Cancel", hộp sẽ trả về null. Ví dụ muốn người dùng nhập tên:

```
let ten = prompt("Tên","Tên của bạn");
```



Hình 6-4 Hộp thoại nhập

Javascript cho phép thực thi mã theo các khoảng thời gian xác định. Những khoảng thời gian này được gọi là sự kiện thời gian. Các phương thức chính để sử dụng với JavaScript là setTimeout(), setInterval() và clearTimeout().

Phương thức **setTimeout**() thực thi một hàm, sau khi chờ một số mili giây xác định, ví dụ thực hiện lệnh in chữ "Hiển thị thông tin" **sau 1000 mili giây**:

```
const thoigian = setTimeout(function (){
    alert("Hiện thị thông tin")
}, 1000);
```

setInterval(): giống như setTimeout (), nhưng lặp lại việc thực hiện phương thức liên tục. Ví dụ thực hiện lệnh cứ **sau mỗi1000 mili giây**.

```
const thoigian = setTimeout(function (){
    console.log("Hiện thị thông tin")
}, 1000);
```

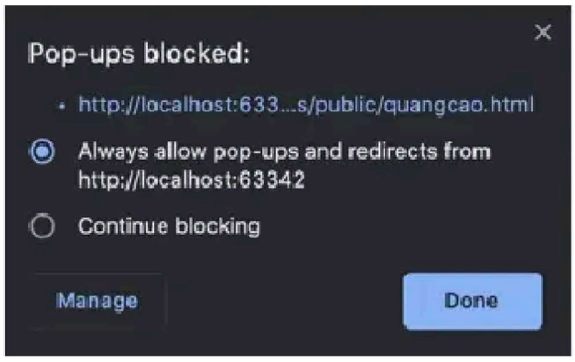
Phương thức **clearTimeout**() dừng thực hiện các sự kiện thời gian đã thiết lập, ví dụ:

```
clearTimeout(thoigian)
```

Phương thức **open()** và **close()** dùng mở một cửa sổ mới, hoặc đóng cửa sổ hiện tại. Ví dụ mở một cửa sổ mới hiển thị file **quangcao.html**, 3 giây sau thì đóng cửa sổ đó lại:

```
var quangcao = open("quangcao.html", "", "width=600, height=400")
setTimeout(function () {
    quangcao.close()
}, 3000);
```

Tuy nhiên cần chú ý, trình duyệt phải cho phép bật Popup thì câu lệnh open() mới có tác dụng:



Hình 6-5 Bật Popup trên trình duyệt Chrome

Phương thức **moveBy()** và **moveTo()** dùng để di chuyển cửa sổ. Ví dụ di chuyển cửa sổ mới mở (open) ra:

```
var quangcao = open("quangcao.html", "", "width=600, height=400")
quangcao.moveTo(100, 100); //Di chuyển đến vị trí (100,100)
quangcao.moveBy(50, 50);   //Di chuyển đến vị trí mới cách vị trí ban đầu 50 về
bên phải và 50 xuống phía dưới
```

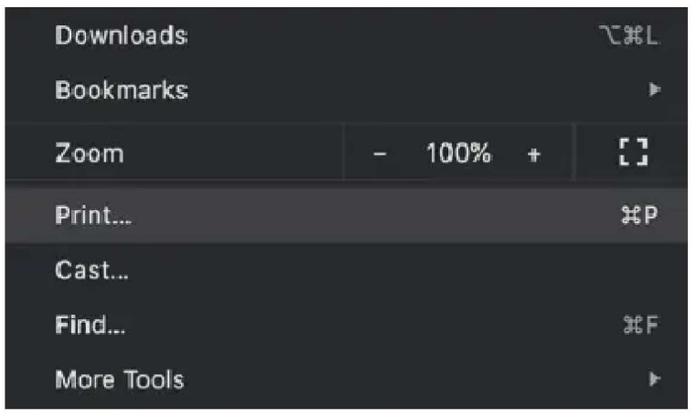
Phương thức **resizeBy()** và **resizeTo()** dùng để thay đổi kích thước của của sổ. Ví dụ thay đổi kích thước của cửa sổ mới mở (open) ra:

```
var quangcao = open("quangcao.html", "", "width=600, height=400")
quangcao.resizeTo(300, 300);  //Thay đổi kích thước thành 300px x 300px
quangcao.resizeBy(100, 100);  //Thay đổi kích thước tăng thêm 100px mỗi chiều
```

Phương thức **scrollBy()** và **scrollTo()** dùng để cuộn cửa sổ. Ví dụ cuộn cửa sổ hiện tại:

```
scrollTo(0, 100);  //Cuộn theo trục y (Cuộn dọc) đến vị trí 50
scroolBy(0, 50);   //Cuộn theo trục tung (Cuọn dọc) đến vị trí cách vị trí ban
đầu 50
```

Phương thức **print()** dùng để **in nội dung** của cửa sổ. Tác dụng của hàm này tương tự như khi người dùng chọn **Print...** từ trình duyệt:



Hình 6-6 Tùy chọn Print của trình duyệt

Bên cạnh các phương thức trên window còn có nhiều các phương thức hữu ích nữa. Và quan trọng hơn đối tượng window có rất nhiều thuộc tính quan trọng. Phần dưới sẽ trình bày về các thuộc tính này.

6.1.3 Các thuộc tính chứa các đối tượng quan trọng

Từ đối tượng **window**, có thể truy cập vào nhiều thuộc tính, đó chính là các đối tượng quan trọng, trong phần này sẽ tìm hiểu các đối tượng này.

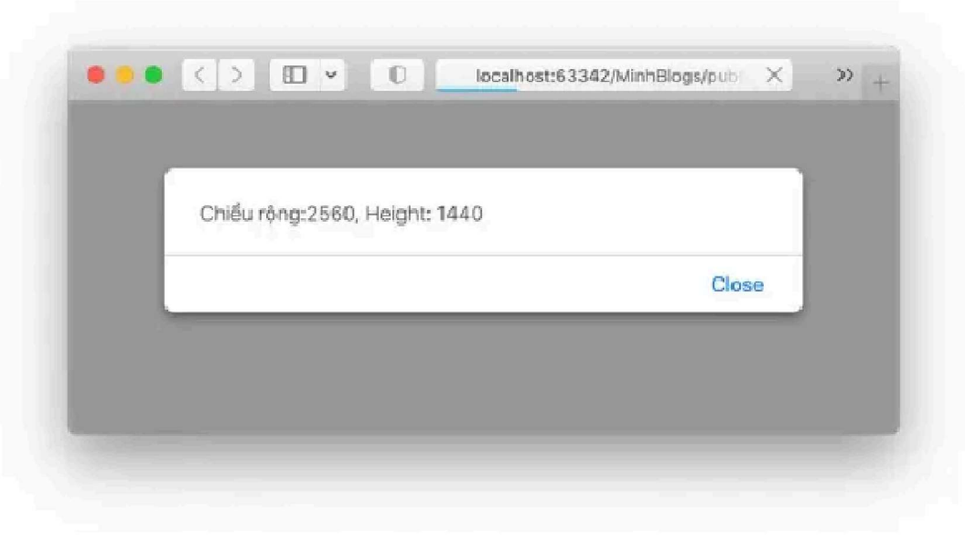
Đối tượng màn hình (screen) chứa các thông tin về màn hình của người dùng. Một số thuộc tính quan trọng của đối tượng này bao gồm:

- **width**: chiều rộng màn hình.
- **height**: chiều cao màn hình.
- **availWidth**: chiều rộng khả dụng của màn hình.
- **availHeight**: chiều cao khả dụng của màn hình.
- **colorDepth**: Độ sâu màu màn hình.
- **isExtend**: có màn hình ngoài hay không.
- **orientation**: màn hình xoay dọc hay xoay ngang.

Thuộc tính **width** và **height** cho biết kích thước của màn hình, trong khi **availWidth** và **availHeight** cho biết kích thước **"khả dụng"** của màn hình, đây là kích thước có thể sử dụng cho trình duyệt. Ví dụ để in ra kích thước "khả dụng" của màn hình của người dùng, có thể viết mã như sau:

```
alert("Chiều rộng:" + screen.availWidth + ", Chiều cao: " + screen.availHeight)
```

Kết quả:



Hình 6-7 Hiển thị kích thước khả dụng của màn hình

Đối tượng vị trí (location) được sử dụng để lấy địa chỉ của Web hiện tại hoặc điều hướng trình duyệt sang hiển thị một trang Web khác. Một số thuộc tính thông dụng của đối tượng này bao gồm:

- **href**: trả về url của trang hiện tại.
- **hostname**: trả về tên miền của trang hiện tại.
- **pathname**: trả về đường dẫn và tên file của trang hiện thại.
- **protocol:** trả về giao thức của trang hiện tại (ví dụ http, https).
- **origin**: trả về tên miền của trang hiện tại.

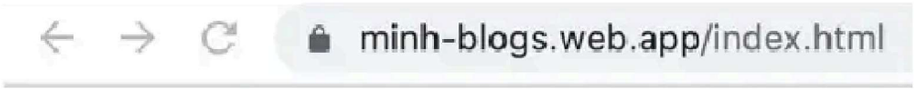
Ví dụ Website được triển khai lên với tên miền là **https://minh-blogs.Web.app** thì các giá trị trả thuộc tính của đối tượng **location** này sẽ như sau:

Hình 6-8 Các thuộc tính giá trị của đối tượng location

Đối tượng lịch sử (history) chứa lịch sử duyệt trang của trình duyệt. Ví dụ người dùng đang ở trang của mình nhưng trước đó đã ghé thăm 5 trang thì **history.length** sẽ trả về 5. Vì lý do bảo vệ quyền riêng tư của người dùng thì sẽ không thể biết được đó là 5 trang nào. Tuy nhiên các phương thức của **history** sẽ phép điều hướng tới các trang đó bằng cá phương thức sau:

- **back()** - quay lại trang trước đó.
- **forward()** - vào trang sau đó (sau khi back).
- **go()** - Di chuyển đến trang bất kỳ.

Có thể nói thuộc tính history có các phương thức cho phép điều hướng Website giống như thanh điều hướng trên trình duyệt (thường có 2 nút **back** và **forward**).



Hình 6-9 Thanh điều hướng trên trình duyệt

Đối tượng trình duyệt (navigator) được sử dụng để xác định thông tin trình duyệt. Một số thuộc tính quan trọng có thể kể đến như:

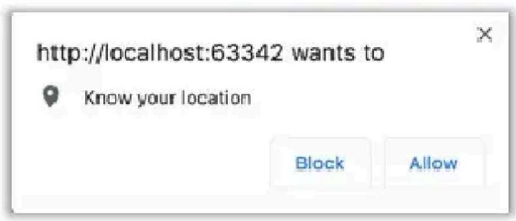
- **appName**: tên ứng dụng của trình duyệt.
- **appCodeName**: tên mã ứng dụng của trình duyệt.
- **appVersion**: thông tin phiên bản về trình duyệt.
- **language**: ngôn ngữ của trình duyệt.
- **cookieEnabled**: trả về true nếu cookie được bật.
- **geolocation**: trả về vị trí địa lý của người dùng.

Chú ý, thông tin từ đối tượng **navigator** thường có thể bị sai lệch vì các trình duyệt khác nhau có thể sử dụng cùng tên, ngoài ra, dữ liệu này có thể được thay đổi bởi chủ sở hữu trình duyệt.

Thuộc tính **geolocation** trả về vị trí địa lý của người dùng nếu người dùng cho phép truy cập. Khi đó, phương thức **getCurrentPosition()** được sử dụng để trả vị trí của người dùng. Ví dụ dưới đây trả về vĩ độ và kinh độ của vị trí người dùng:

```
if (navigator.geolocation) {
    navigator.geolocation.getCurrentPosition(function (position) {
        let vitri = "Vĩ độ: " + position.coords.latitude + " Kinh độ: " +
position.coords.longitude;
        console.log(vitri)
    });
} else {
    console.log("Không thể truy cập vào vị trí cu người dùng")
}
```

Người dùng sẽ được hỏi có cấp quyền truy cập cho Website hay không, nếu người dùng chấp thuận khi đó mới có thể gọi hàm **getCurrrentPosition()**.



Hình 6-10 Yêu cầu truy cập vị trí của người dùng

Đối tượng cookieStore cho phép lưu trữ thông tin người dùng trong các trang Web. Cookies được lưu trong các cặp **tên/giá trị**. Khi người dùng truy cập một trang Web, tên của người dùng có thể được lưu trữ trong một cookie. Lần tới khi người dùng truy cập trang, cookie sẽ "nhớ" tên của người dùng và đưa ra một lời chào.

Có thể **tạo cookie** bằng cách sử dụng **set** như sau, khi gán thường khai báo thêm thuộc tính **exprires** xác định thời điểm mà cookie tồn tại:

```
cookieStore.set({
    name: "ten",
    value: "Phạm Dương Minh",
    expires: Date.now() + 24 * 60 * 60 * 1000 // Một ngày
})
```

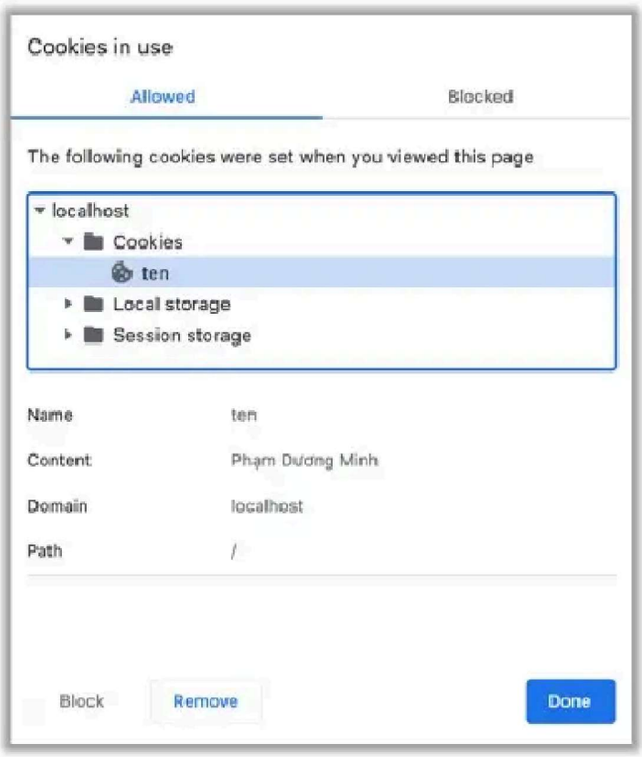
Cookie có thể được **đọc** bằng cách truy cập trực tiếp vào cookie:

```
let ten = cookieStore.get("ten")
```

Có thể **xóa cookie** bằng cách sử dụng phương thức delete():

```
cookieStore.delete("ten")
```

Cookie có thể dễ dàng được xem từ trình duyệt:



Hình 6-11 Xem cookie bằng trình duyệt

Đối tượng lưu localStorage được sử dụng để lưu trữ dữ liệu tương tự như cookie trên máy người dùng nhưng không có ngày hết hạn, và có dung lượng lưu trữ cao hơn nhiều. Để lưu dữ

liệu sử dụng phương thức **setItem().** Ngoài ra cũng có thể gán giá trị cho **localStorage** như là một thuộc tính của đối tượng này:

```
localStorage.setItem("ten", "Phạm Dương Minh")
localStorage.tuoi = 22
```

Để lấy dữ liệu từ localStorage sử dụng phương thức getItem hoặc truy cập trực tiếp vào tên thuộc tính:

```
console.log(localStorage.getItem("ten"))
console.log(localStorage.tuoi)
```

Phương thức **removeItem()** được sử dụng để xóa dữ liệu **localStorage,** ngoài ra cũng có thể gán cho thuộc tính đó bằng rỗng. Ví dụ:

```
localStorage.removeItem("ten")
localStorage.tuoi = null
```

Đối tượng lưu trữ sessionStorage tương tự với đối tượng localStorage, ngoại trừ việc nó lưu trữ dữ liệu chỉ trong một phiên. Dữ liệu sẽ bị xóa khi người dùng đóng tab trình duyệt cụ thể. Ví dụ sau đây đếm số lần người dùng đã nhấp vào nút, trong phiên hiện tại:

```
if (sessionStorage.solanbam) {
    sessionStorage.solanbam = Number(sessionStorage. solanbam) + 1;
} else {
    sessionStorage.solanbam = 1;
}
document.getElementById("result").innerHTML = "Bạn đã bấm " +
sessionStorage.solanbam + " lần trong phiên làm việc này";
```

6.2 ĐỐI TƯỢNG TÀI LIỆU

Trong phần trên chúng ta đã tìm hiểu các thuộc tính quan trọng của window. Đó cũng là các đối tượng được sử dụng rất nhiều trong khi lập trình ở phía máy khách (client). Ngoài các đối tượng đó ra, thì **đối tượng tài liệu (document)** cũng là một trong những thuộc tính của **window**. Đối tượng này định nghĩa toàn bộ các thông tin về **tài liệu HTML**. Từ đó có thể viết mã JavaScript để tương tác với các phần tử trong một tài liệu HTML như:

- Thay đổi các phần tử HTML.
- Thay đổi các thuộc tính HTML.
- Thay đổi các định dạng (CSS) của các phần tử HTML.
- Xóa/thêm các phần tử/ thuộc tính HTML.
- Tạo các sự kiện mới trong trang.

Nói cách khác, DOM là một tiêu chuẩn cho việc nhận, thay đổi, thêm hoặc xóa các phần tử HTML. Ví dụ sau thay đổi nội dung của phần tử **<p>** có id là "heading":

```
<html>
<body>
 <p id="welcome"></p>
 <script>
```

```
   document.getElementById("welcome").innerHTML = "Chào buổi sáng";
 </script>
</body>
</html>
```

Cách phổ biến nhất để truy cập phần tử HTML là sử dụng id của phần tử. Trong ví dụ trên, phương thức **getElementById()** được sử dụng để tìm phần tử có id bằng "demo", trong khi **innerHTML** là một thuộc tính cần thay đổi.

6.2.1 Tìm phần tử HTML

Để có thể thay đổi được các phần tử HTML, điều quan trọng là cần phải xác định được các phần tử đó, đối tượng tài liệu (document) cung cấp các phương thức để tìm phần tử đó. Ví dụ để truy xuất tới một phần tử HTML theo ID ta sử dụng phương thức getElementById() như sau:

```
var phần-tử = document.getElementById('id-của-phần-tử');
```

Phương thức **getElementsByTagName()** được dùng để tìm phần tử HTML theo tên. Tên phần tử HTML chính là tên các thẻ như p, a, div, ... Và ta sẽ truy xuất tới nó bằng cú pháp sau:

```
var phần-tử = document.getElementsByTagName('tên-phần-tử');
```

Ví dụ:

```
<html>
<body>
 <input type="text" id="name" value="test">
 <input type="password" id="password">
 <script>
   var element = document.getElementsByTagName('input');
   document.write(element[0].value);
 </script>
</body>
</html>
```

Trong một trang Web có thể có nhiều thẻ HTML giống nhau (ví dụ có hai thẻ div) nên phương thức getElementsByTagName() sẽ trả về một mảng các object chứ không phải là một object nữa, chính vì vậy ta sẽ lấy input thứ nhất nên truyền số 0 vào.

Để tìm các phần tử HTML có theo tên lớp (class) sử dụng phương thức **getElementsByClassName():**

```
var phần-tử = document.getElementsByClassName('tên-lớp');
```

Ví dụ:

```
// Lấy thẻ input
var element = document.getElementsByClassName('input');
// Lấy giá trị của thẻ input
document.write(element[0].value);
```

Tương tự như tìm theo tên phần tử HTML thì tìm theo tên class sẽ trả về một mảng các đối tượng nên sẽ phải sử dụng cú pháp truy xuất mảng để chọn đúng đối tượng muốn lấy.

Phương thức **querySelectorAll()** giúp tìm phần tử theo **bộ chọn CSS** (đã học trong chương trước). Ví dụ để tìm tất cả các phần tử **div có class là note**:

```
var element = document.querySelectorAll("div.note");
```

6.2.2 Thay đổi thuộc tính

Để thay đổi giá trị của một thuộc tính phần tử HTML bất kì, sử dụng cú pháp sau:

```
element.attributeName = "new value";
```

Ví dụ, để lấy và thay đổi nội dung bên trong một thẻ HTML, truy cập vào thuộc tính innerHTML:

```
document.getElementById("content").innerHTML = "<h1>Nội dung</h1>";
```

Viết chương trình để khi click vào một **button** thì chuyển nó thành **textbox**:

```
<html>
<body>
 <script language="JavaScript ">
   function change() {
     var object = document.getElementById("object");
     var type = object.type;
     object.type = 'text';
   }
 </script>
 <input type="button" value="Click here" onclick="change()" id="object" />
</body>
</html>
```

style bản chất nó cũng là một thuộc tính của các phần tử HTML. Thuộc tính style này sẽ chứa tất cả các thuộc tính của CSS, như vậy để thay đổi thuộc tính style, cần viết với cú pháp:

```
document.getElementById("object").style.cssName = 'something';
```

Chú ý, trường hợp thuộc tính có dấu gạch ngang như: font-size, line-height, margin-bottom thì thuộc tính đó trong style sẽ có tên là fontSize, lineHeight, marginBottom, nghĩa là sẽ bỏ đi dấu gạch ngang và viết hoa ký tự đầu tiên của chữ thứ hai. Lưu ý rằng có những thuộc tính nếu chưa thiết lập CSS cho nó thì khi lấy giá trị sẽ là một giá trị rỗng.

6.3 XỬ LÝ SỰ KIỆN

6.3.1 Gắn sự kiện vào DOM

JavaScript có thể được thực thi khi xảy ra sự kiện, Ví dụ như khi người dùng nhấp vào phần tử HTML:

```
<!DOCTYPE html>
<html>
<body>
 <h1 onclick="changeText(this)">Click on this text!</h1>
 <script>
```

```
   function changeText(id) {
     id.innerHTML = "Ooops!";
   }
 </script>
</body>
</html>
```

Một sự kiện là một hành động nào đó tác động lên các phần tử HTML. Trong ví dụ trên, JavaScript sẽ thay đổi nội dung của phần tử HTML có id là demo.

Ví dụ về các sự kiện HTML:

- Khi người dùng nhấp chuột.
- Khi một trang Web đã được tải.
- Khi một hình ảnh đã được tải.
- Khi chuột di chuyển qua một phần tử.
- Khi một trường đầu vào được thay đổi.
- Khi một biểu mẫu HTML được gửi.
- Khi người dùng vuốt một phím.

Sự kiện	**Mô tả**
onchange	Khi một phần tử HTML thay đổi
onclick	Khi người dùng bấm lên một phần tử HTML
onmouseover	Khi người dùng di chuyển chuột trên một phần tử HTML
onmouseout	Khi người dùng di chuyển chuột khỏi một phần tử HTML
onkeydown	Khi người dùng bấm bàn phím
onload	Khi trình duyệt kết thúc việc tải trang

Các sự kiện onload và onunload được kích hoạt khi người dùng vào hoặc rời trang. Sự kiện onload này có thể được sử dụng để kiểm tra loại trình duyệt và phiên bản trình duyệt của khách truy cập và tải phiên bản phù hợp của trang Web dựa trên thông tin. Các sự kiện onload và onunload có thể được sử dụng để tải (hoặc ghi) cookie. Sự kiện onchange thường được sử dụng để kiểm tra dữ liệu người dùng nhập. Các sự kiện onmouseover và onmouseout có thể được sử dụng để kích hoạt một chức năng khi người dùng di chuột vào và ra khỏi một phần tử HTML.

Phương thức addEventListener() gắn một "sự kiện" cho một phần tử HTML. Cú pháp:

```
element.addEventListener(event, function, useCapture);
```

- Tham số đầu tiên là loại sự kiện (như "click" hoặc "mousedown" hoặc bất kỳ Sự kiện DOM DOM nào khác.)
- Tham số thứ hai là phương thức chúng ta muốn gọi khi sự kiện xảy ra.
- Tham số thứ ba là một giá trị **boolean** xác định xem nên sử dụng cơ chế **bubbling** hoặc **capturing** sự kiện. Tham số này là tùy chọn.

Ví dụ lắng nghe sự kiện khi người dùng nhấp vào một phần tử HTML (Ví dụ nút bấm):

```
<!DOCTYPE html>
<html>
```

```
<body>
 <button id="myBtn">Click on this text!</h1>
 <script>
   document.getElementById("myBtn").addEventListener("click", displayDate);
   function displayDate() {
     document.getElementById("myBtn").innerHTML = new Date();
   }
 </script>
</body>
</html>
```

Phương thức addEventListener() gắn một "sự kiện" cho một phần tử HTML mà không ghi đè lên xử lý sự kiện hiện có. Chính vì vậy mà chúng ta có thể thêm nhiều trình xử lý sự kiện vào một phần tử. Cần lưu ý rằng có thể lắng nghe sự kiện vào bất kỳ đối tượng DOM nào, không chỉ các phần tử HTML (Ví dụ đối tượng window). Khi sử dụng addEventListener(), mã JavaScript được tách ra khỏi mã HTML, điều này giúp mã dễ đọc hơn và cho phép thêm trình lắng nghe sự kiện ngay cả khi không kiểm soát file HTML (Ví dụ không thể sửa file HTML).

Ví dụ hiển thị thông báo "Hello World" khi người dùng nhấp vào một phần tử HTML:

```
var element = document.getElementById("myBtn")
element.addEventListener("click", function () { alert("Hello World!"); });
```

6.3.2 Lan truyền sự kiện

Lan truyền sự kiện là một cách xác định thứ tự phần tử khi một sự kiện xảy ra. Nếu có phần tử <p> bên trong phần tử và người dùng nhấp vào phần tử <p>, thì sự kiện "nhấp chuột" của phần tử nào sẽ được xử lý trước?

Có hai cách truyền bá sự kiện trong HTML DOM, **bubbling** và **capturing**.

- Nếu sử dụng kiểu **bubbling**, sự kiện phần tử bên trong được xử lý trước và sau đó là bên ngoài: sự kiện nhấp chuột của phần tử <p> được xử lý trước, sau đó là sự kiện nhấp chuột của phần tử .
- Nếu sử dụng kiểu **capturing**, sự kiện bên ngoài của hầu hết các phần tử được xử lý trước và sau đó bên trong: sự kiện nhấp chuột của phần tử sẽ được xử lý trước, sau đó là sự kiện nhấp chuột của phần tử <p>.

Với phương thức addEventListener(), có thể chỉ định loại lan truyền bằng cách sử dụng tham số "useCapture":

```
element.addEventListener(event, function, useCapture);
```

Giá trị mặc định là false, sẽ sử dụng lan truyền **bubbling**, khi giá trị được đặt thành true, sự kiện sử dụng lan truyền **capturing**. Ví dụ muốn sử dụng kiểu lan truyền **capturing:**

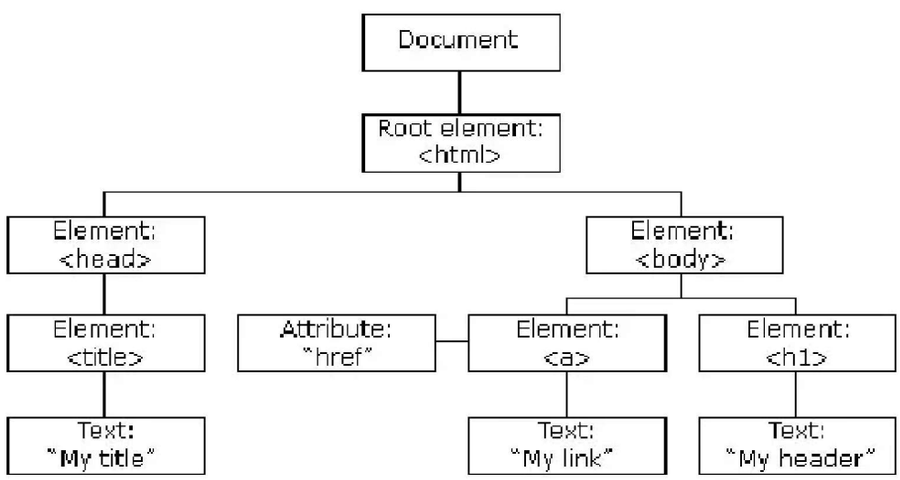
```
document.getElementById("myP").addEventListener("click", myFunction, true);
document.getElementById("myDiv").addEventListener("click", myFunction, true);
```

6.4 NÚT (NODE)

Theo tiêu chuẩn DOM W3C HTML, mọi thứ trong tài liệu HTML là một nút (node). Như vậy có một số node được gọi là phần tử HTML, số còn lại thì không.

- Toàn bộ tài liệu là một nút tài liệu.
- Toàn bộ document là 1 nút lớn.
- Mỗi phần tử HTML là một nút phần tử.
- Văn bản bên trong các phần tử HTML là nút văn bản.
- Mỗi bình luận là một nút bình luận.

Một tài liệu HTML sẽ có thể biểu diễn là một cây gồm nhiều nút như sau:



Với HTML DOM, tất cả các nút trong cây nút có thể được truy cập bằng JavaScript .

Các nút mới có thể được tạo và tất cả các nút có thể được sửa đổi hoặc xóa.

6.4.1 Mối quan hệ nút

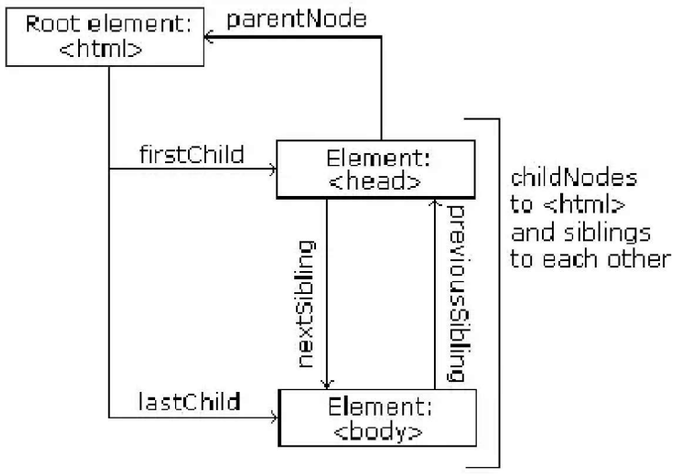
Các nút có quan hệ với nhau:

- Các nút trong cây nút có mối quan hệ phân cấp với nhau.
- Các thuật ngữ cha (parent), con cái (child) và anh chị em (sibling) được sử dụng để mô tả các mối quan hệ.
- Trong cây nút, nút trên cùng được gọi là nút gốc.
- Mỗi nút có chính xác một cha, ngoại trừ gốc (không có cha).
- Một nút có thể có một số nút con (child).
- Anh chị em (sibling) là các nút có cùng cha mẹ.

Từ HTML ở trên,có thể đọc:

- <html> là nút gốc, không có nút cha
- <html>là cha của <head> và <body>
- <head> là con đầu của <html>
- <body> là con cuối của <html>

- <head> có một con: <title>
- <title> có một con (một nút văn bản): "My Title"
- <body>có hai con: <a> và <h1>
- <h1> có một con: "My header"
- <p> có một con: "My link"
- <h1>và <a>là anh chị em



Có thể sử dụng các thuộc tính nút sau để điều hướng giữa các nút:

- parentNode
- childNodes[nodenumber]
- firstChild
- lastChild
- nextSibling
- previousSibling

6.4.2 Thuộc tính nodeName

Thuộc tính nodeName xác định tên của một nút.

- nodeName chỉ đọc
- nodeName của một nút phần tử giống như tên thẻ
- nodeName của một nút thuộc tính là tên thuộc tính
- nodeName của một nút văn bản luôn là #text
- nodeName của nút tài liệu luôn là #document

6.4.3 Thuộc tính nodeValue

Thuộc tính nodeValue xác định giá trị của một nút.

- nodeValue cho các nút phần tử là null
- nodeValue cho các nút văn bản là chính văn bản
- nodeValue cho các nút thuộc tính là giá trị thuộc tính

6.4.4 Thuộc tính nodeType

Các thuộc tính nodeType chỉ được đọc. Nó trả về kiểu của một nút.

6.4.5 Tạo các phần tử HTML mới (Nút)

Để thêm một phần tử mới vào HTML DOM, trước tiên phải tạo phần tử (nút phần tử), sau đó nối nó vào phần tử hiện có. Ví dụ:

```
<div id="div1">
 <p id="p1">This is a paragraph.</p>
 <p id="p2">This is another paragraph.</p>
</div>
<script>
 var para = document.createElement("p");
 para.innerHTML = "This is a new paragraph"
 var element = document.getElementById("div1");
 element.appendChild(para);
</script>
```

Tạo các phần tử HTML vào đầu, sử dụng phương thức insertBefore(). Ví dụ:

```
<div id="div1">
 <p id="p1">This is a paragraph.</p>
 <p id="p2">This is another paragraph.</p>
</div>
<script>
 var para = document.createElement("p");
 var node = document.createTextNode("This is new paragraph");
 para.appendChild(node);
 var element = document.getElementById("div1");
 var child = document.getElementById("p1");
 element.insertBefore(para, child);
</script>
```

6.4.6 Xóa các phần tử HTML hiện có

Để xóa phần tử HTML, sử dụng remove(). Ví dụ:

```
<div>
 <p id="p1">This is a paragraph.</p>
 <p id="p2">This is another paragraph.</p>
</div>
<script>
var elmnt = document.getElementById("p1");
elmnt.remove();
</script>
```

Đối với các trình duyệt không hỗ trợ phương thức remove(), phải tìm nút cha để xóa phần tử. Ví dụ:

```
<div id="div1">
 <p id="p1">This is a paragraph.</p>
 <p id="p2">This is another paragraph.</p>
</div>
```

```
<script>
var parent = document.getElementById("div1");
var child = document.getElementById("p1");
parent.removeChild(child);
</script>
```

6.5 XÁC THỰC BIỂU MẪU

HTML forms được sử dụng để người dùng nhập liệu, và thông thường được kiểm tra lại bằng JavaScript . Ví dụ có một form:

```
<form name="myForm" action="/action_page.php" onsubmit="return validateForm()"
method="post">
 Name: <input type="text" name="fname">
 <input type="submit" value="Submit">
</form>
```

Mã JavaScript sẽ chạy và kiểm tra khi người dùng submit form:

```
<script>
 function validateForm() {
   if (document.forms["myForm"]["fname"].value == "") {
     alert("Name must be filled out");
     return false;
   }
 }
</script>
```

JavaScript cung cấp 2 phương thức là checkValidity() và setCustomValidity() để kiểm tra sự hợp lệ của dữ liệu. Hai phương thức này sẽ kết hợp với các thuộc tính ràng buộc dữ liệu liệt kê trong bảng sau:

Thuộc tính	**Mô tả**
disabled	Đối tượng nhập liệu không thể sử dụng
max	Giá trị lớn nhất cho phép
min	Giá trị nhỏ nhất cho phép
pattern	Khuôn mẫu của dữ liệu
required	Bắt buộc nhập dữ liệu
type	Kiểu dữ liệu

Khi các thành phần nhập liệu được quy định các thuộc tính trên, trong phương thức kiểm tra dữ liệu với JavaScript chỉ cần gọi phương thức checkValidity(). Nếu dữ liệu hợp lệ sẽ trả về đúng (true), hoặc không hợp lệ sẽ trả về sai (false) từ đó là cơ sở để đưa ra các cảnh báo hoặc nhắc nhở cần thiết. Đối với các kiểu dữ liệu phức tạp hơn, có thể kết hợp sử dụng thêm các thuộc tính của thuộc tính validity.

Các thuộc tính của thuộc tính validity được liệt kê trong bảng dưới đây.

Thuộc tính	Ý nghĩa
customError	Bằng true nếu được thiết lập cảnh báo tùy chỉnh
patternMismatch	Bằng true nếu dữ liệu nhập sai với khuôn mẫu
rangeOverflow	Bằng true nếu dữ liệu lớn hơn giá trị cho phép
rangeUnderflow	Bằng true nếu dữ liệu nhỏ hơn giá trị cho phép
stepMismatch	Bằng true nếu dữ liệu sai thuộc tính bước (step)
tooLong	Bằng true nếu giá trị vượt quá chiều dài cho phép
typeMismatch	Bằng true dữ liệu bị sai kiểu
valueMissing	Bằng true nếu chưa có dữ liệu
valid	Bằng true nếu dữ liệu là hợp lệ

Ví dụ sau minh họa cách sử dụng thuộc tính trên để kiểm tra dữ liệu người dùng nhập có nằm trong khoảng cho phép (từ 20 đến 100) hay không.

```
Number: <input id="id1" type="number" max="100" min="20">
<input type="submit" value="Submit" onClick="myFunction()">
<p id="demo"></p>
<script>
 function myFunction() {
   var txt = "";
   if (document.getElementById("id1").validity.rangeOverflow) {
     txt = "Value too large";
   }
   else if (document.getElementById("id1").validity.rangeUnderflow) {
     txt = "Value too small";
   }
   document.getElementById("demo").innerHTML = txt;
 }
</script>
```

6.6 KÉO/ THẢ

Kéo và **thả** **(Drag** and **Drop)** là một tính năng rất phổ biến. Đó là khi "chọn" một vật thể và kéo nó đến một vị trí khác. Trong HTML5 bất kỳ phần tử HTML nào cũng có thể Drag được. Ví dụ dưới đây là một ví dụ kéo và thả đơn giản:

6.6.1 Tạo một phần tử có thể kéo

Đầu tiên, để làm cho một phần tử có thể kéo được, hãy đặt thuộc tính **draggable** thành **true**:

```
<div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)"
 style="width: 100px;height: 100px; border:1px solid #aaaaaa;"></div>
<br>
<h1 id="drag1" draggable="true" ondragstart="drag(event)">Text </h1>
```

6.6.2 Kéo - ondragstart và setData()

Sau đó, chỉ định những gì sẽ xảy ra khi phần tử được kéo. Thuộc tính ondragstart gọi một phương thức drag(ev) chỉ định dữ liệu để thiết lập kiểu dữ liệu và giá trị của dữ liệu kéo:

```
function drag(ev) {
  ev.dataTransfer.setData("text", ev.target.id);
}
```

6.6.3 ondragover

Sự kiện ondragover chỉ định nơi dữ liệu được kéo có thể được thả. Để cho phép thả, chúng ta phải ngăn việc xử lý phần tử mặc định. Điều này được thực hiện bằng cách gọi phương thức event.preventDefault() cho sự kiện ondragover:

```
function allowDrop(ev) {
  ev.preventDefault();
}
```

6.6.4 Thả - ondrop

Khi dữ liệu kéo được thả, một sự kiện thả xảy ra. Trong ví dụ trên, thuộc tính ondrop gọi phương thức drop (event):

```
function drop(ev) {
  ev.preventDefault();
  var data = ev.dataTransfer.getData("text");
  ev.target.appendChild(document.getElementById(data));
}
```

Giải thích:

- Gọi preventDefault() để ngăn việc xử lý dữ liệu mặc định của trình duyệt.
- Lấy dữ liệu được kéo bằng phương thức dataTransfer.getData(). Phương thức này sẽ trả về bất kỳ dữ liệu nào được đặt cùng loại trong phương thức setData().
- Dữ liệu được kéo là id của phần tử được kéo ("drag1").
- Nối phần tử được kéo vào phần tử drop.

Mã nguồn hoàn chỉnh:

```
<div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)"
 style="width: 100px;height: 100px; border:1px solid #aaaaaa;"></div>
<br>
<h1 id="drag1" draggable="true" ondragstart="drag(event)">Text </h1>

<script>
 function allowDrop(ev) {
   ev.preventDefault();
 }
 function drag(ev) {
   ev.dataTransfer.setData("text", ev.target.id);
 }
 function drop(ev) {
```

```
    ev.preventDefault();
    var data = ev.dataTransfer.getData("text");
    ev.target.appendChild(document.getElementById(data));
  }
</script>
```

6.7 CANVAS

Phần tử **<canvas>** HTML được sử dụng để vẽ đồ họa. Phần tử <canvas> chỉ là một container để hiển thị các đối tượng đồ họa. Phải sử dụng JavaScript để thực sự vẽ đồ họa. Canvas có một số phương thức để vẽ đường, hộp, vòng tròn, văn bản và thêm hình ảnh. Canvas là một khu vực hình chữ nhật trên trang HTML. Theo mặc định, một khung vẽ không có viền và không có nội dung. Lưu ý, luôn chỉ định một thuộc tính id (sẽ được đề cập trong tập lệnh) và width/height để xác định kích thước của khung vẽ. Để thêm một đường viền, sử dụng style thuộc tính.

Dưới đây là một ví dụ về một khung vẽ cơ bản, trống rỗng:

```
<canvas id="myCanvas" width="200" height="100" style="border:1px solid
#000000;">
</canvas>
```

Vẽ đường thẳng

```
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.moveTo(0, 0);
ctx.lineTo(200, 100);
ctx.stroke();
```

Vẽ một vòng tròn

```
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.beginPath();
ctx.arc(95, 50, 40, 0, 2 * Math.PI);
ctx.stroke();
```

Vẽ một văn bản

```
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.font = "30px Arial";
ctx.fillText("Hello World", 10, 50);
```

Stroke Text

```
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.font = "30px Arial";
ctx.strokeText("Hello World", 10, 50);
```

Vẽ Gradient tuyến tính

```
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
```

```
// Create gradient
var grd = ctx.createLinearGradient(0, 0, 200, 0);
grd.addColorStop(0, "red");
grd.addColorStop(1, "white");
// Fill with gradient
ctx.fillStyle = grd;
ctx.fillRect(10, 10, 150, 80);
```

Vẽ Gradient tròn

```
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
// Create gradient
var grd = ctx.createRadialGradient(75, 50, 5, 90, 60, 100);
grd.addColorStop(0, "red");
grd.addColorStop(1, "white");
// Fill with gradient
ctx.fillStyle = grd;
ctx.fillRect(10, 10, 150, 80);
```

Vẽ hình ảnh

```
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
var img = document.getElementById("scream");
ctx.drawImage(img, 10, 10);
```

6.8 JSON

6.8.1 JSON là gì?

JSON là viết tắt của **J**ava**S**cript **O**bject **N**otation. Là một định dạng trao đổi, lưu trữ dữ liệu. Cú pháp JSON có nguồn gốc từ cú pháp ký hiệu đối tượng JavaScript, nhưng định dạng JSON chỉ là văn bản. Mã để đọc và tạo dữ liệu JSON có thể được viết bằng bất kỳ ngôn ngữ lập trình nào. Định dạng JSON hoàn toàn giống với mã để tạo các đối tượng JavaScript. Do sự giống nhau này, một chương trình JavaScript có thể dễ dàng chuyển đổi dữ liệu JSON thành các đối tượng JavaScript gốc.

Cú pháp của JSON có quy tắc sau:

- Dữ liệu nằm trong cặp tên/giá trị.
- Dữ liệu được phân tách bằng dấu phẩy.
- Dấu ngoặc nhọn khai báo đối tượng.
- Dấu ngoặc vuông khai báo mảng.

Ví dụ một file JSON

```
{
"employees":[
  {"firstName":"John", "lastName":"Doe"},
  {"firstName":"Anna", "lastName":"Smith"},
  {"firstName":"Peter", "lastName":"Jones"}
```

```
]
}
```

Cách sử dụng phổ biến của JSON là đọc dữ liệu từ máy chủ Web sau đó sử dụng hàm JavaScript JSON.parse() để chuyển thành đối tượng JavaScript:

```
let text = '{ "employees" : [' +
'{ "firstName":"John" , "lastName":"Doe" },' +
'{ "firstName":"Anna" , "lastName":"Smith" },' +
'{ "firstName":"Peter" , "lastName":"Jones" } ]}';
let obj = JSON.parse(text);
```

Hiển thị dữ liệu:

```
<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML =
obj.employees[1].firstName + " " + obj.employees[1].lastName;
</script>
```

Nếu có dữ liệu được lưu trữ trong một đối tượng, có thể chuyển đổi các đối tượng thành một chuỗi JSON, và gửi nó đến máy chủ:

```
var myObj = {name: "John", age: 31, city: "New York"};
var myJSON = JSON.stringify(myObj);
window.location = "demo_json.php?x=" + myJSON;
```

JSON làm cho nó có thể lưu trữ các đối tượng JavaScript dưới dạng văn bản.

```
// Storing data:
myObj = {name: "John", age: 31, city: "New York"};
myJSON = JSON.stringify(myObj);
localStorage.setItem("testJSON", myJSON);

// Retrieving data:
text = localStorage.getItem("testJSON");
obj = JSON.parse(text);
document.getElementById("demo").innerHTML = obj.name;
```

6.8.2 JSON và XML

Cả JSON và XML đều có thể được sử dụng với công dụng tương tự nhau. Tuy nhiên cú pháp khác nhau. Ví dụ dưới đây JSON và XML cùng được sử dụng để lưu trữ thông tin của 3 nhân viên. Khi đó JSON sẽ được biểu diễn như sau:

```
{"employees":[
 { "firstName":"John", "lastName":"Doe" },
 { "firstName":"Anna", "lastName":"Smith" },
 { "firstName":"Peter", "lastName":"Jones" }
]}
```

XML sẽ được biểu diễn như sau:

```
<employees>
 <employee>
  <firstName>John</firstName> <lastName>Doe</lastName>
 </employee>
 <employee>
  <firstName>Anna</firstName> <lastName>Smith</lastName>
 </employee>
 <employee>
  <firstName>Peter</firstName> <lastName>Jones</lastName>
 </employee>
</employees>
```

JSON giống XML: Con người đều có thể đọc hiểu được. Đều lưu trữ dữ liệu theo kiểu phân cấp. Đều có thể phân tích và được sử dụng bởi rất nhiều ngôn ngữ lập trình. **JSON không giống XML**: JSON không sử dụng thẻ kết thúc, JSON ngắn hơn, JSON đọc và ghi nhanh hơn, JSON có thể sử dụng mảng. **Tại sao JSON tốt hơn XML**: XML khó phân tích hơn JSON, JSON có cú pháp tương tự như đối tượng JS, Đối với các ứng dụng AJAX, JSON nhanh hơn và dễ dàng hơn XML.

6.8.3 Kiểu dữ liệu JSON

Trong JSON, giá trị phải là một trong các kiểu dữ liệu Chuỗi, Số, Đối tượng (đối tượng JSON), Mảng, Boolean hoặc giá trị Rỗng (null). Giá trị JSON không thể là một hàm giống như đối tượng JavaScript.

Chuỗi trong JSON phải được viết trong dấu ngoặc kép. Ví dụ:

```
{ "name":"John" }
```

Số trong JSON phải là số nguyên hoặc một dấu chấm động:

```
{ "age":30}
```

Giá trị trong JSON có thể là các đối tượng, Đối tượng như các giá trị trong JSON phải tuân theo các quy tắc tương tự như các đối tượng JSON:

```
{
"employee":{ "name":"John", "age":30, "city":"New York" }
}
```

Giá trị trong JSON có thể là mảng. Cú pháp của mảng là sử dụng dấu ngoặc vuông:

```
{
"employees":[ "John", "Anna", "Peter" ]
}
```

Giá trị trong JSON có thể đúng/sai (boolean):

```
{ "sale":true }
```

Giá trị trong JSON có thể rỗng (null):

```
{ "middlename":null }
```

6.9 AJAX

6.9.1 AJAX là gì?

AJAX (Asynchronous JavaScript And XML) là một công cụ để từ phía trình duyệt có thể:

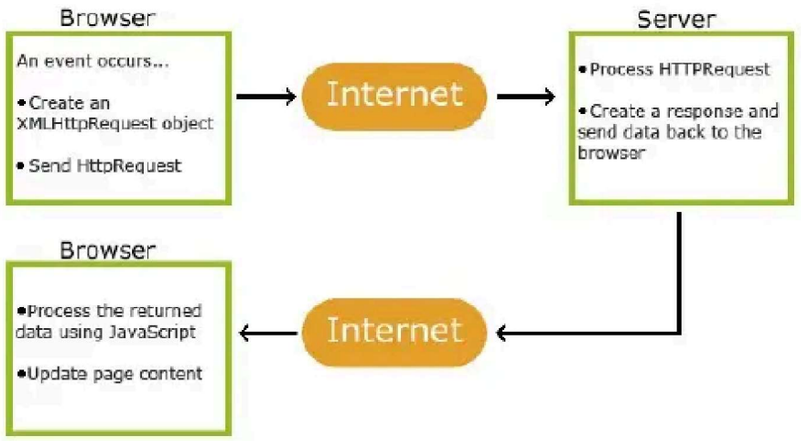
- Đọc dữ liệu từ một máy chủ Web – sau khi trang đã được tải về.
- Cập nhật một trang Web mà không cần tải lại trang.
- Gửi dữ liệu đến một máy chủ Web – ở chế độ nền.

AJAX là một cái tên dễ gây hiểu nhầm. Ứng dụng AJAX có thể sử dụng XML để vận chuyển dữ liệu, nhưng nó thường chuyển từ văn bản gốc sang văn bản JSON. AJAX cho phép các trang Web cập nhật không đồng bộ bằng cách trao đổi dữ liệu với một máy chủ Web đằng sau hậu trường. Điều này có nghĩa rằng nó có thể cập nhật các phần của một trang Web, mà không cần tải lại toàn bộ trang.

Ví dụ

```
<!DOCTYPE html>
<html>
<body>
 <div id="demo">
   <h2>Let AJAX change this text</h2>
   <button type="button" onclick="loadData()">Change Content</button>
 </div>
 <script>

   function loadData() {
     var xhttp = new XMLHttpRequest();
     xhttp.onreadystatechange = function () {
       if (this.readyState == 4 && this.status == 200) {
         document.getElementById("demo").innerHTML = this.responseText;
       }
     };
     xhttp.open("GET", "https://myWebsite-3db42-default-
rtdb.firebaseio.com/Website.json", true);
     xhttp.send();
   }
 </script>
</body>
</html>
```

AJAX làm việc như thế nào?



1. Sự kiện xảy ra trong một trang Web. Một đối tượng **XMLHttpRequest** được tạo.
2. Đối tượng **XMLHttpRequest** gửi một yêu cầu đến một máy chủ Web.
3. Máy chủ xử lý yêu cầu.
4. Máy chủ sẽ gửi một hồi đáp trở lại trang Web.
5. Dữ liệu trả về được xử lý dùng để cập nhật cho trang Web.

6.9.2 Các bước để thực hiện AJAX

Bước 1: Tạo một đối tượng XMLHttpRequest :

```
variable = new XMLHttpRequest();
```

Bước 2: Gửi một yêu cầu đến máy chủ. Để gửi yêu cầu đến một máy chủ, ta dùng phương thức open() và send() của đối tượng XMLHttpRequest :

```
xhttp.open("GET", "ajax_info.txt", true);
xhttp.send();
```

Một yêu cầu POST đơn giản :

```
xhttp.open("POST", "demo_post.asp", true);
xhttp.send();
```

Để dữ liệu POST như một HTML form, thêm một tiêu đề HTTP với setRequestHeader(). Xác định các dữ liệu muốn gửi trong phương thức send() :

```
xhttp.open("POST", "ajax_test.asp", true);
xhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
xhttp.send("fname=Henry&lname=Ford");
```

Với đối tượng XMLHttpRequest có thể định nghĩa một hàm được thực thi khi yêu cầu nhận được một câu trả lời. Hàm được định nghĩa trong thuộc tính onreadystatechange của đối tượng XMLHttpRequest :

```
xhttp.onreadystatechange = function() {
 if (this.readyState == 4 && this.status == 200) {
  document.getElementById("demo").innerHTML = this.responseText;
 }
};
xhttp.open("GET", "ajax_info.txt", true);
xhttp.send();
```

Các thuộc tính status và các thuộc tính statusText giữ trạng thái của đối tượng XMLHttpRequest.

Thuộc tính	Miêu tả
readyState	Giữ trạng thái của XMLHttpRequest. 0: yêu cầu không được khởi tạo 1: kết nối máy chủ được thiết lập 2: yêu cầu được nhận 3: xử lý yêu cầu 4: kết thúc yêu cầu và phản hồi sẵn sàng
status	200: "OK" 403: "Forbidden" 404: "Page not found"
statusText	Trả lại tráng thái văn bản (e.g. "OK" or "Not Found")

Các hàm onreadystatechange được gọi mỗi khi thay đổi readyState. Khi readyState bằng 4 và status bằng 200, phản hồi đã sẵn sàng :

```
function loadDoc() {
 var xhttp = new XMLHttpRequest();
 xhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
   document.getElementById("demo").innerHTML =
   this.responseText;
  }
 };
 xhttp.open("GET", "ajax_info.txt", true);
 xhttp.send();
}
```

Một hàm callback là một hàm thông qua như là một tham số tới một hàm khác:

```
loadDoc("url-1", myFunction1);
loadDoc("url-2", myFunction2);
function loadDoc(url, cFunction) {
 var xhttp;
 xhttp = new XMLHttpRequest();
 xhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
```

```
   cFunction(this);
  }
 };
 xhttp.open("GET", url, true);
 xhttp.send();
}
function myFunction1(xhttp) {
 // action goes here
}
function myFunction2(xhttp) {
 // action goes here
}
```

6.9.3 Các thuộc tính và phương thức

Thuộc tính Server Response:

Thuộc tính	Mô tả
responseText	Lấy dữ liệu phản hồi dưới dạng chuỗi
responseXML	Lấy dữ liệu phản hồi dưới dạng dữ liệu XML

Phương thức Server Response:

Phương thức	Mô tả
getResponseHeader()	Trả về thông tin tiêu đề cụ thể từ tài nguyên máy chủ.
getAllResponseHeaders()	Trả về tất cả thông tin tiêu đề từ tài nguyên máy chủ.

Thuộc tính responseText trả về phản hồi máy chủ dưới dạng một chuỗi JavaScript, vàcó thể sử dụng nó cho phù hợp :

```
document.getElementById("demo").innerHTML = xhttp.responseText;
```

Đối tượng XMLHttpRequest có trình phân tích cú pháp XML dựng sẵn.

```
xmlDoc = xhttp.responseXML;
txt = "";
x = xmlDoc.getElementsByTagName("ARTIST");
for (i = 0; i < x.length; i++) {
 txt += x[i].childNodes[0].nodeValue + "<br>";
}
document.getElementById("demo").innerHTML = txt;
xhttp.open("GET", "cd_catalog.xml", true);
xhttp.send();
```

Phương thức **getAllResponseHeaders()** trả về tất cả thông tin tiêu đề từ phản hồi của máy chủ.

```
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
 if (this.readyState == 4 && this.status == 200) {
  document.getElementById("demo").innerHTML =
```

```
  this.getAllResponseHeaders();
 }
};
```

Phương thức **getResponseHeader()** trả về thông tin tiêu đề cụ thể từ phản hồi của máy chủ.

```
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
 if (this.readyState == 4 && this.status == 200) {
  document.getElementById("demo").innerHTML =
  this.getResponseHeader("Last-Modified");
 }
};
xhttp.open("GET", "ajax_info.txt", true);
xhttp.send();
```

THỰC HÀNH

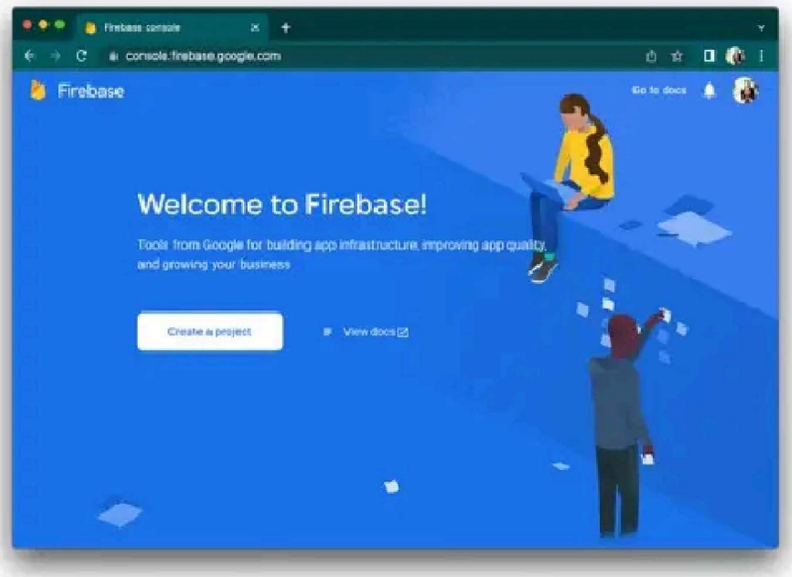
Bài 1. Tạo một đối tượng chuyển động:

```
<!DOCTYPE html>
<html>
<style>
 #container {
   width: 400px;
   height: 400px;
   position: relative;
   background: yellow;
 }

 #animate {
   width: 50px;
   height: 50px;
   position: absolute;
   background-color: red;
 }
</style>

<body>
 <p><button onclick="myMove()">Click Me</button></p>
 <div id="container">
   <div id="animate"></div>
 </div>
 <script>
   function myMove() {
     var elem = document.getElementById("animate");
     var pos = 0;
     var id = setInterval(frame, 5);
     function frame() {
       if (pos == 350) {
         clearInterval(id);
       } else {
         pos++;
         elem.style.top = pos + "px";
         elem.style.left = pos + "px";
```

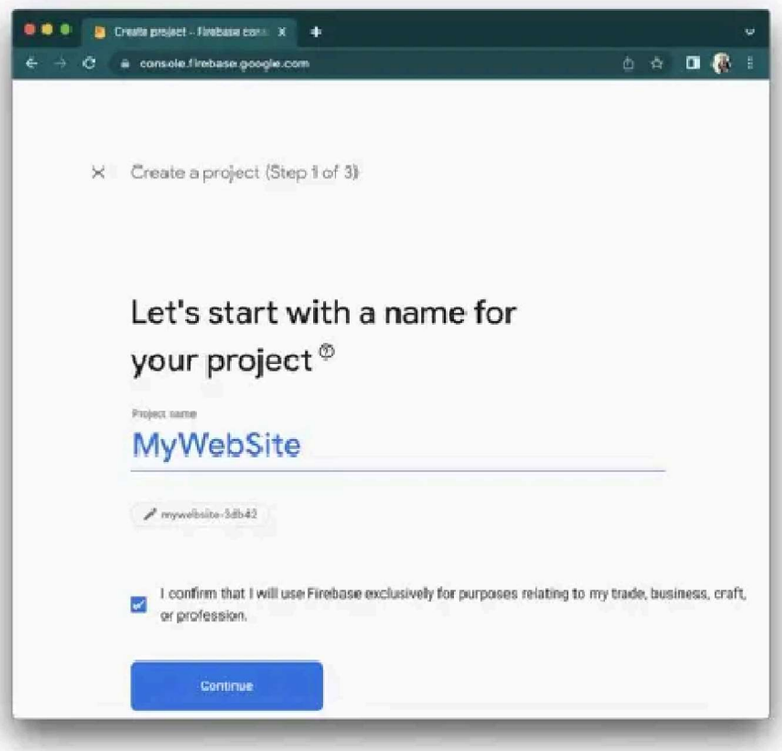
```
      }
    }
  }
 </script>
</body>
</html>
```

Bài 2. Bài thức hành với cơ sở dữ liệu JSON trên Firebase

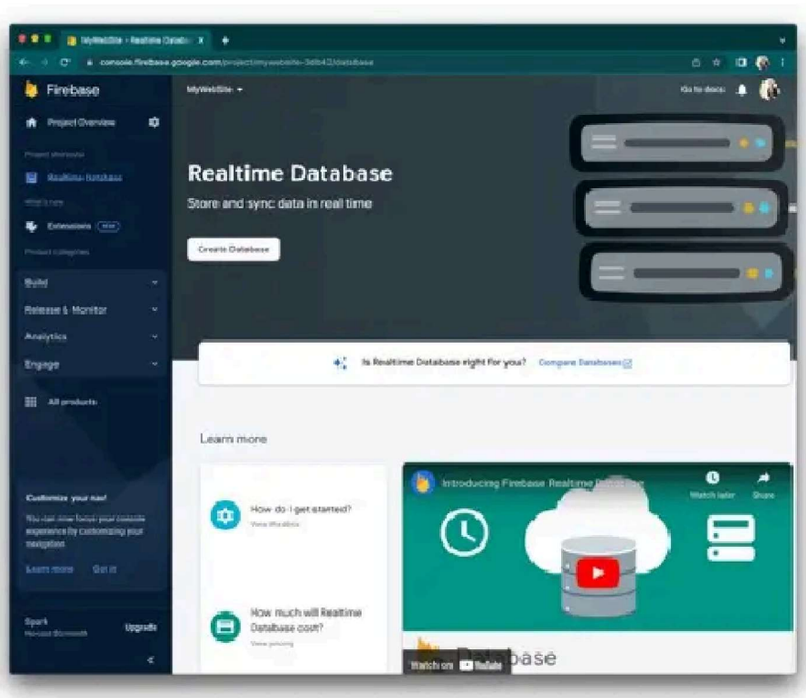
Firebase là gì?



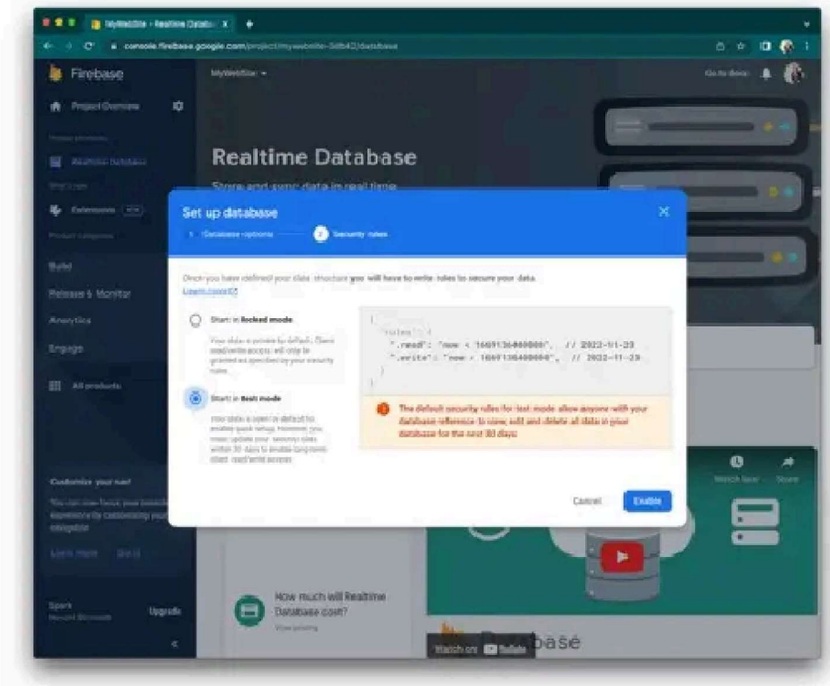
Tạo Project



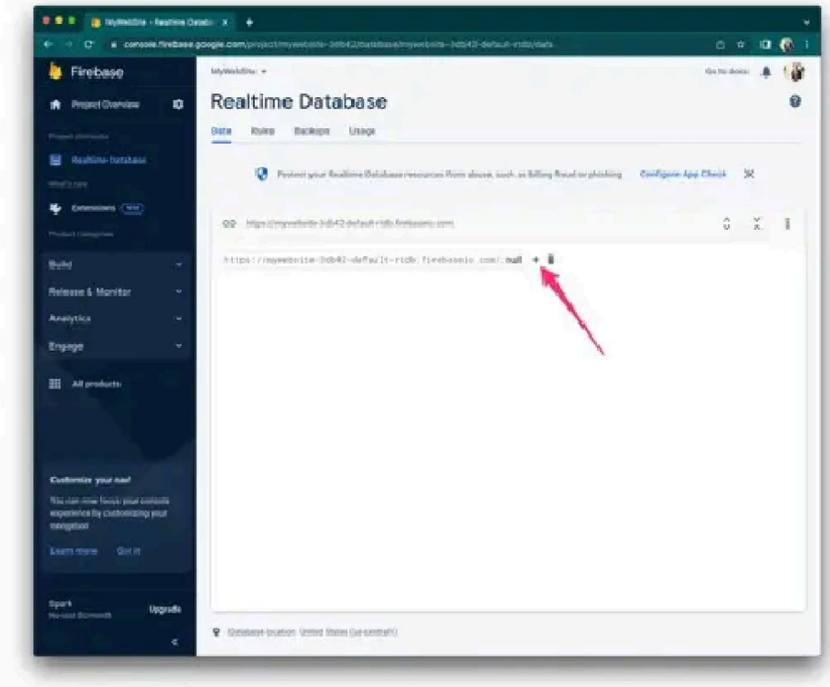
Tạo Database



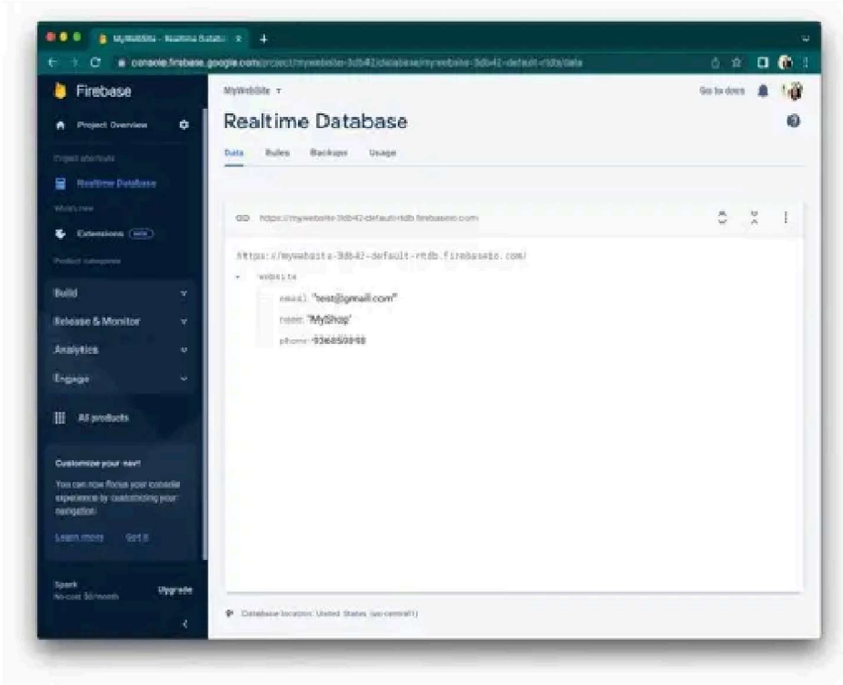
Chọn Test Mode để cấu hình theo thời gian:



Chọn dấu + để thêm dữ liệu JSON



Thêm dữ liệu vào như sau:

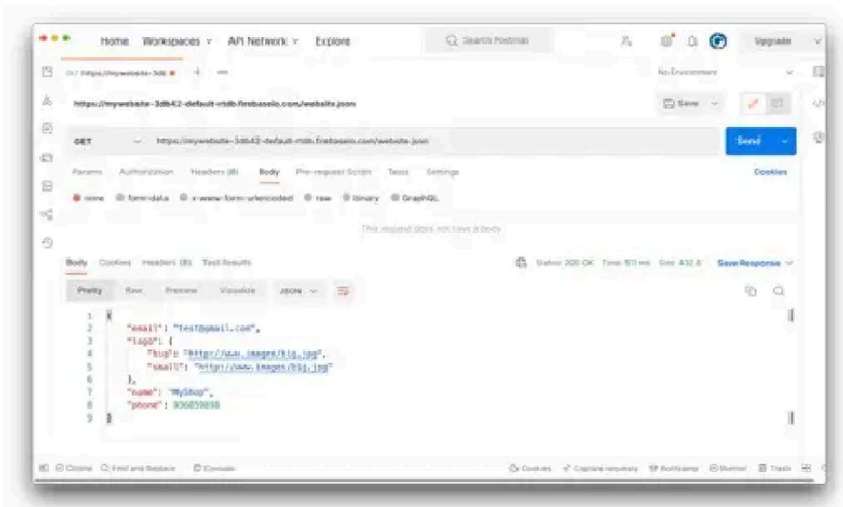


Truy cập vào dữ liệu trên bằng đường dẫn sau:

```
https://myWebsite-3db42-default-rtdb.firebaseio.com/Website.json
```

Bài 3. Sử dụng PostMan

GET

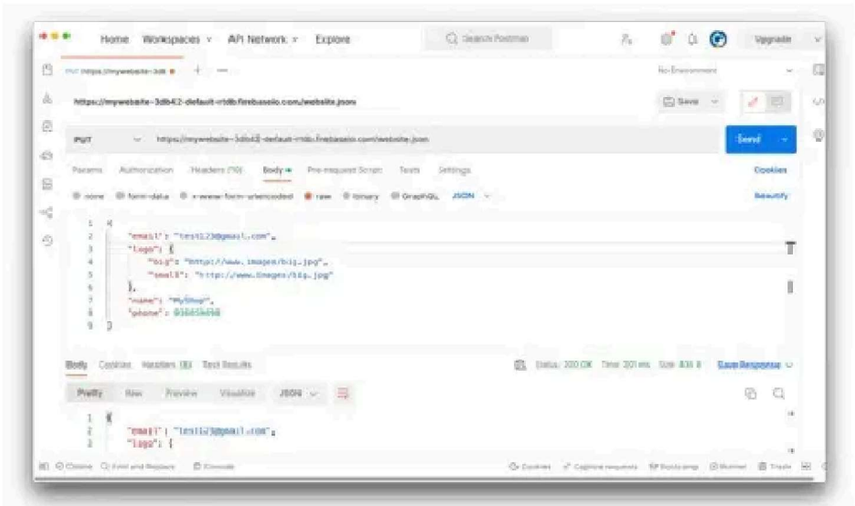


Nội dung file json

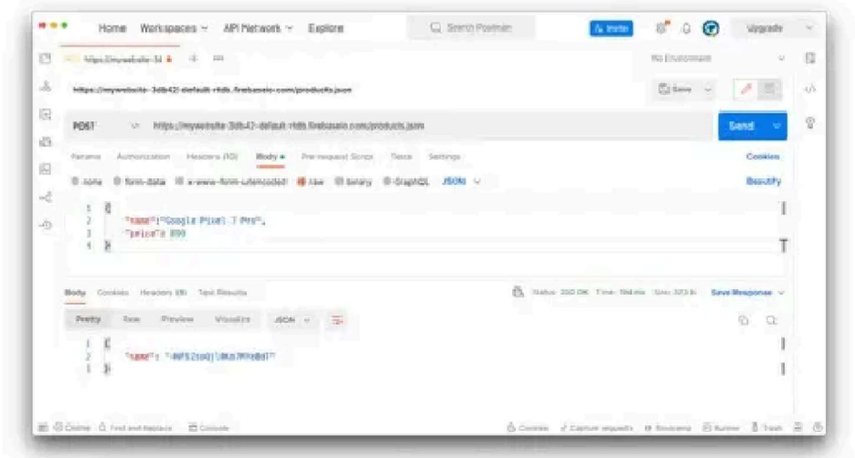
```
{
   "email": "test@gmail.com",
```

```
    "logo": {
        "big": "http://www.images/big.jpg",
        "small": "http://www.images/big.jpg"
    },
    "name": "MyShop",
    "phone": 936859898
}
```

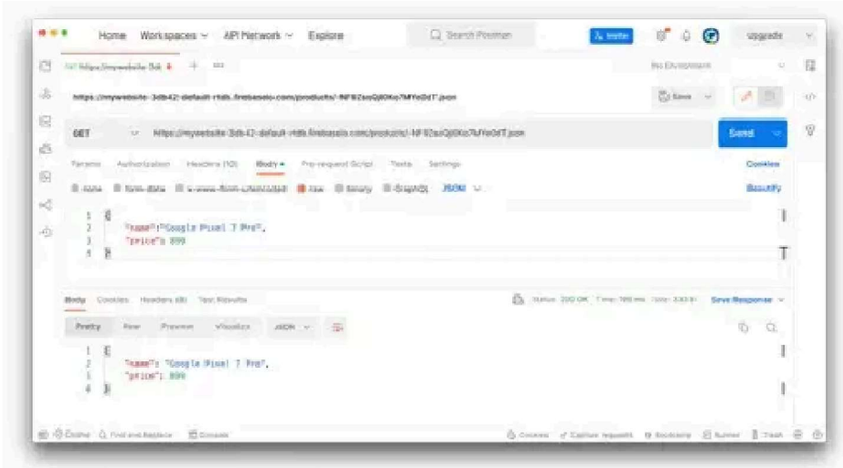
PUT



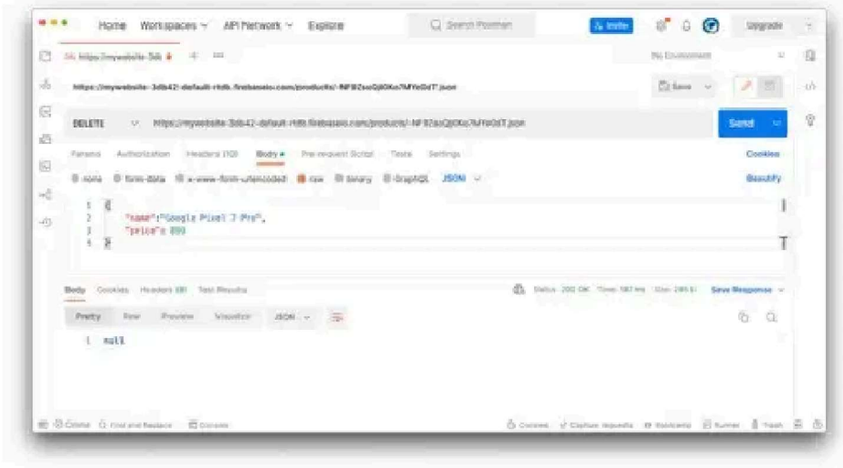
POST



GET 1 ITEM



DELETE



Bài 4. Thực hành thêm xóa dữ liệu trên Firebase

Thêm dữ liệu:

```
<body>
 <div id="demo">
   <h2>Let AJAX add a Product</h2>
   <button type="button" onclick="addProduct()">Add Product</button>
 </div>
 <script>
   function addProduct() {
     var xhttp = new XMLHttpRequest();
     xhttp.onreadystatechange = function () {
       if (this.readyState == 4 && this.status == 200) {
         document.getElementById("demo").innerHTML = this.responseText;
       }
     };
     xhttp.open("POST", "https://myWebsite-3db42-default-
rtdb.firebaseio.com/products.json", true);
     let data = JSON.stringify({ "name": "Samsung S22", "price": 999 })
```

```
    xhttp.send(data);
  }
 </script>
</body>
```

Lấy dữ liệu:

```
<style>
 .product{
   border: 1px solid red
 }
</style>
<body>
 <div id="demo" style="display:flex"></div>
 <script>
   function getProducts() {
     var xhttp = new XMLHttpRequest();
     xhttp.onreadystatechange = function () {
       if (this.readyState == 4 && this.status == 200) {
         let products = JSON.parse(this.responseText);
         for (p in products) {
           const nodeProduct = document.createElement("div");
           nodeProduct.setAttribute("class","product")

           //Name
           const nodeName = document.createElement("div");
           nodeName.appendChild(document.createTextNode(products[p].name));
           nodeProduct.append(nodeName)

           //Price
           const nodePrice = document.createElement("div");
           nodePrice.appendChild(document.createTextNode(products[p].price));
           nodeProduct.append(nodePrice)

           document.getElementById("demo").appendChild(nodeProduct)
         }
       }
     };
     xhttp.open("GET", "https://myWebsite-3db42-default-
rtdb.firebaseio.com/products.json", true);
     xhttp.send();
   }

   getProducts();
 </script>
</body>
```

Bài 5. Đọc và lưu dữ liệu vào Minh Blogs

Tạo thư mục js với file main.js, sử dụng <script> để liên kết tới file js này. Khai báo này nên được viết trước khi đóng thẻ <body>

```
...
<script src="js/main.js"></script>
</body>
```

Tạo dữ liệu trên firebase:

Sửa lại phần section, gán id cho 2 phần tử:

```
<div class="contact">
   <h4 class="contact-title">Hòm thư</h4>
   <a href="mailto:(webmail@gmail.com)" id="info-email"></a>
   <h4 class="contact-title">Điện thoại</h4>
   <a href="tel:+1255-568-6523" id="info-phone"></a>
   <h4 class="contact-title">Địa chỉ</h4>
   <a href="https://www.google.com/maps" target="blank" id="info-address"></a>
</div>
```

Bổ sung mã js:

```
function getInfo() {
   var xhttp = new XMLHttpRequest();
   xhttp.onreadystatechange = function () {
       if (this.readyState == 4 && this.status == 200) {
           let info = JSON.parse(this.responseText);
           document.getElementById("info-email").innerText = info.email;
           document.getElementById("info-phone").innerText = info.phone;
           document.getElementById("info-address").innerText = info.address;
       }
   };
   xhttp.open("GET", "https://minh-blogs-default-rtdb.firebaseio.com/info.json",
true);
   xhttp.send();
}
getInfo();
```

Kiểm tra kết quả đảm bảo dữ liệu đã được tải:

Tương tự, hãy viết hàm getArticles() để lấy các bài viết từ firebase và hiển thị trên trang Bài viết

Để lưu dữ liệu lên Firebase (*), Hãy viết một hàm:

```
function saveFormData() {
   var xhttp = new XMLHttpRequest();
   xhttp.onreadystatechange = function () {
       if (this.readyState == 4 && this.status == 200) {
           document.getElementById("register-form").innerHTML = "Done!";
       }
   };
   xhttp.open("POST", "https://minh-blogs-default-
rtdb.firebaseio.com/register.json", true);
   xhttp.send();
}
```

Hãy sửa hàm trên, để khi bấm submit thì gọi hàm này và đẩy dữ liệu lên Firebase.

CÂU HỎI ÔN TẬP LÝ THUYẾT

1. Mã lệnh JavaScript được đặt bên trong phần tử HTML nào?

A. <script>

B. <js>

C. <javascript>

D. <scripting>

2. Cú pháp JavaScript chính xác để thay đổi nội dung của phần tử HTML * This is a demonstration. * là gì?

A. #demo.innerHTML = "Hello World!";

B. document.getElementById("demo").innerHTML = "Hello World!";

C. document.getElement("p").innerHTML = "Hello World!";

D. document.getElementByName("p").innerHTML = "Hello World!";

3. Đâu là vị trí chính xác để chèn JavaScript?

A. Phần <body>

B. Cả phần <head> và phần <body> đều đúng

C. Phần <head>

D. Phần <javascrip>

4. Cú pháp chính xác để tham chiếu đến external script có tên "xxx.js" là gì?

A. <script href="xxx.js">

B. <script name="xxx.js">

C. <script src="xxx.js">

D. <script url="xxx.js">

5. Làm thế nào để viết "Hello World" trong một hộp cảnh báo (alert box)?

A. alertBox("Hello World");

B. alert("Hello World");

C. msgBox("Hello World");

D. msg("Hello World");

6. Sự kiện nào xảy ra khi người dùng nhấp vào một phần tử HTML?

A. Onmouseclick

B. Onmouseover

C. Onclick

D. onchange

7. Đối tượng window có chứa những thuộc tính và phương thức nào để tương tác với các thành phần trên trang web?

A. document

E. navigator

F. screen

G. tất cả đều đúng

8. Sử dụng lệnh nào để lấy phần tử HTML trong tài liệu HTML

A. document.getValueById()

B. document.getElementByName()

C. document.getElementById()

D. document.getElementByClassName()

9. JSON viết tắt của từ gì?

A. JavaScript Object Node

B. JavaScript Object Notation

C. JavaScript Object Naming

D. JavaScript Object Numbering

10. Cú pháp JSON bao gồm những ký tự nào?

A. Dấu ngoặc đơn {}

B. Dấu ngoặc vuông []

C. Dấu ngoặc nhọn ()

D. Cả A và B

11. Các giá trị JSON có thể là kiểu dữ liệu gì?

A. Number

B. Boolean

C. String

D. Tất cả đều đúng

12. Sử dụng phương thức nào để chuyển đổi một chuỗi JSON thành một đối tượng JavaScript?

A. SON.parse()

B. JSON.stringify()
C. JSON.convert()
D. JSON.load()

13. AJAX có thể truyền dữ liệu giữa trang web và máy chủ dưới định dạng nào?

A. XML
B. JSON
C. Cả A và B đều đúng
D. Tất cả các phương án đều sai

14. Phương thức nào của đối tượng XMLHttpRequest cho phép gửi yêu cầu?

A. open()
B. send()
C. setRequestHeader()
D. connect()

15. Để kiểm tra trạng thái của yêu cầu AJAX trong JavaScript, sử dụng thuộc tính nào của đối tượng XMLHttpRequest?

A. status
B. responseText
C. readyState
D. responseStatus

BÀI TẬP TỰ THỰC HÀNH

Bài 1. Xử lý biểu mẫu

Tạo một trang web đơn giản với một ô input và một nút submit. Khi người dùng nhập một chuỗi và nhấn vào nút submit, sử dụng Javascript để hiển thị chuỗi đó lên trang web. Kiểm tra và đưa ra thông báo nếu người dùng không nhập.

Bài 2. Website giới thiệu sản phẩm

Tạo một trang web với một danh sách các sản phẩm. Khi người dùng nhấp vào một sản phẩm trong danh sách, sử dụng Javascript để hiển thị thông tin chi tiết về sản phẩm đó. Thông tin chi tiết có thể được lấy từ một chuỗi JSON.

Bài 3. Tạo một trò chơi đơn giản

Tạo một trò chơi đơn giản, ví dụ một trò chơi đố vui với nhiều câu hỏi. Sử dụng Javascript để hiển thị câu hỏi và các phương án trả lời cho người chơi, và kiểm tra xem người chơi đã chọn đáp án đúng hay không cho mỗi phương án trả lời.

TÀI LIỆU THAM KHẢO

[1] Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics (2018), Jennifer Robbins, O'Reilly Media

[2] JavaScript: The Definitive Guide: Master the World's Most-Used Programming Language (2020), David Flanagan, O'Reilly Media

[3] Professional JavaScript® for Web Developers (2020), Matt Frisbie, Wrox

CHƯƠNG 7: CÁC CÔNG CỤ HỖ TRỢ

Bên cạnh HTML và CSS, các nhà phát triển Web thường sử dụng các công cụ phát triển Web được cung cấp bởi bên thứ ba nhằm tăng tính hiệu quả cũng như năng suất trong công việc. Các công cụ này có thể là một plugin giúp tạo mã HTML tự động từ các từ viết tắt, đến một framework cho phép phát triển Web được nhanh chóng hơn. Ngoài ra, người học cũng được tìm hiểu một số công cụ của bên thứ ba phổ biến nhằm xây dựng một trang Web. Đây cũng sẽ là cơ sở để người học có thể đưa ra những lựa chọn phù hợp về công nghệ cũng như giải pháp cho việc phát triển Web trong tương lai.

7.1 EMMET

Emmet là một bộ plugin dành cho trình soạn thảo văn bản cho phép mã hóa và chỉnh sửa tốc độ cao ở định dạng HTML, XML, XSLT và các định dạng mã có cấu trúc khác thông qua hỗ trợ nội dung. Emmet giúp viết và làm việc với mã trong trình chỉnh sửa dễ dàng hơn. Emmet hỗ trợ cho các nhà phát triển web, giúp tăng tốc độ viết mã HTML và CSS, thay vì phải gõ thủ công từng đoạn mã một. Các mã viết tắt được sử dụng trong Emmet được gọi là "snippets", và chúng bao gồm các thuộc tính và giá trị phổ biến, cũng như các phần tử HTML thường được sử dụng.

Ví dụ muốn tạo một nav bên trong có thẻ ul và 5 thẻ li, trong mỗi thẻ li có một liên kết được đặt id và có nội dung kèm theo thì khi sử dụng Emmet sẽ viết bằng cú pháp viết tắt như sau:

```
nav>ul>li*5>a[href= “target.html”]#item${Link Text}
```

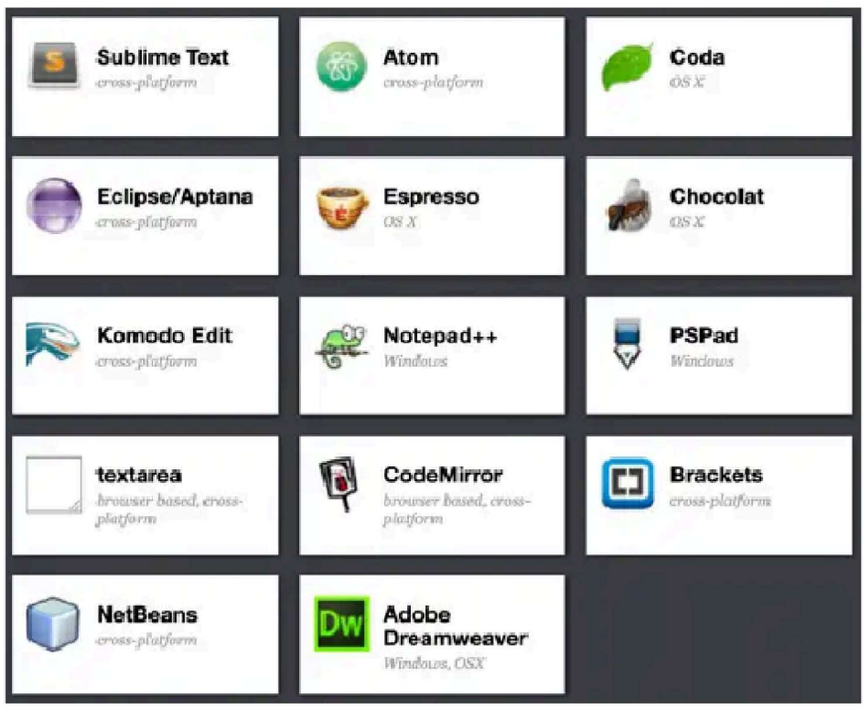
Mã HTML sẽ tự động được tạo ra như sau:

```
<nav>
<ul>
<li><a href = “target.htm1” id= “item1”>Link Text</a></li>
<li><a href = “target.html” id= “item2”>Link Text</a></li>
<li><a href = “target.html” id= “item3”>Link Text</a></li>
<li><a href = “target.html” id= “item4”>Link Text</a></li>
<li><a href = “target.html” id= “item5”>Link Text</a></li>
</ul>
</nav>
```

Trong các phần tử **<a>**, có thể thấy rằng thuộc tính **href** được đặt thành **"target.html"**. Tuy nhiên, lưu ý rằng ngay cả khi không chỉ định thuộc tính **href** cho phần tử **<a>**, Emmet sẽ tạo thuộc tính này vì đây là t**huộc tính bắt buộc**. Sau đó, nhà phát triển chỉ cần đặt giá trị cho thuộc tính. Điều này tương tự với các thuộc tính bắt buộc khác.

Ngoài thuộc tính **href**, một thuộc tính **id** có giá trị **"item"** và chỉ số được thêm vào mỗi phần tử **<a>** để giá trị của thuộc tính **id** khác nhau đối với mỗi mục. Cuối cùng, văn bản cho mỗi phần tử <a> được đặt thành **"Link Text"**. Đây chỉ là một ví dụ về cách Emmet có thể tăng tốc mã hóa HTML. Nếu thử nghiệm với một số lệnh khác, nhà phát triển sẽ hiểu rõ hơn về cách Emmet có thể trợ giúp và tăng tốc trong việc viết mã nguồn.

Emmet này hỗ trợ hoạt động trên nhiều nền tảng khác nhau:



Hình 7-1 Các công cụ có thể cài Emmet

Sau khi cài đặt **Emmet**, công cụ chứa các lệnh của Emmet có sẵn để sử dụng khi viết mã. Sau đó, có thể sử dụng các lệnh này để thực hiện các hành động được hiển thị như trong bảng. Ngoài ra, có thể sử dụng các phím tắt hiển thị ở đây. Ví dụ: nếu muốn chọn thẻ mở, thuộc tính hoặc giá trị thuộc tính tiếp theo, có thể nhấn Ctrl+Shift+. thay vì chọn lệnh **Emmet > Select Next Item.**

Một số cú pháp viết tắt do Emmet cung cấp cũng được trình bày ở trong bảng tiếp theo. Cú pháp sử dụng này để tạo mã HTML một cách nhanh chóng. Ví dụ, để thêm HTML bắt đầu cho một trang Web, có thể nhập ! rồi chọn Emmet > Expand Abbreviation hoặc nhấn Ctrl+Alt+Enter. Trong một số trình chỉnh sửa, có thể nhấn phím Tab để mở rộng từ viết tắt.

Bảng 7-1 Các thao tác phổ biến trong Emmet

Phím tắt	Tác dụng
Ctrl+Shift+A	Kết thúc HTML đã chọn trong phần tử chỉ định.
Ctrl+/	Thêm một bình luận.
Ctrl+Shift+.	Chọn thẻ mở, thuộc tính hoặc giá trị thuộc tính tiếp theo.
Ctrl+Shift+,	Chọn thẻ mở, thuộc tính hoặc giá trị thuộc tính trước đó.
Ctrl+Shift+k	Xóa phần tử tại vị trí hiện tại của con trỏ, nhưng không xóa nội dung của phần tử.
Ctrl+Shift+Enter	Mở rộng một chữ viết tắt.

Bảng 7-2 Bảng cú pháp viết tắt trong Emmet

Cú pháp	Tác dụng

!	Thêm mã bắt đầu cho trang Web, bao gồm khai báo loại tài liệu và phần khung của tài liệu HTML
tag	Thêm một phần tử, cùng với các thuộc tính bắt buộc
tag*number	Thêm một phần tử theo số lần đã chỉ định
tag[attribute=”value”]	Thêm một phần tử với thuộc tính và giá trị được chỉ định
tag1>tag2	Lồng một phần tử bên trong một phần tử khác
tag#id	Thêm phần tử có thuộc tính id được chỉ định
tag#id$	Thêm nhiều phần tử một với id kèm chỉ báo
tag.classname	Thêm phần tử có thuộc tính lớp và tên
tag.classname$	Thêm nhiều phần tử một với tên lớp kèm chỉ báo
tag{text}	Bọc văn bản bằng một thẻ

Với Emmet, cũng có thể sử dụng chữ viết tắt để tạo các thành phần chỉ định. Điều đó có thể bao gồm việc lặp lại một phần tử với số lần đã chỉ định, bao gồm các thuộc tính, phần tử lồng nhau, bao gồm chỉ số trên tên lớp đối với các phần tử được lặp lại và gói văn bản bằng thẻ.

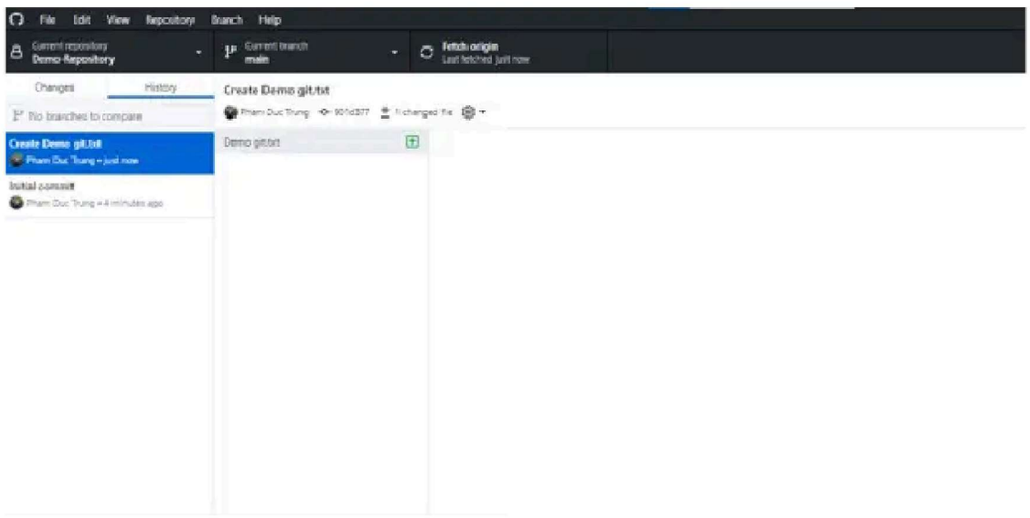
Có thể xem lại tài liệu tài liệu đầy đủ về hướng dẫn sử dụng Emmet tại địa chỉ: **https://docs.emmet.io**.

7.2 Git and Github

Git là một hệ thống **kiểm soát mã nguồn** mà có thể sử dụng để quản lý các dự án được lưu trữ trong các kho mã nguồn. Git giúp cho các thao tác như hoàn tác lỗi, quay lại các phiên bản mã nguồn trước đó, chia sẻ mã nguồn với các nhà phát triển khác mà không tạo ra xung đột... được dễ dàng hơn. Với Git, các dự án được lưu trữ dưới dạng **kho lưu trữ (repository)**. Các kho lưu trữ này có thể chứa các nhánh đại diện cho các trạng thái của dự án theo thời gian. Người dùng có thể điều hướng các nhánh này và sử dụng mã từ bất kỳ nhánh nào để ghi đè các tệp bên trong nhánh chính, hay còn gọi là nhánh chủ, bất kỳ lúc nào.

Git là một ví dụ về hệ thống **kiểm soát mã nguồn Phân tán**. Trong loại hệ thống này, người dùng không cần phải kết nối với máy chủ. Thay vào đó, các thay đổi có thể được thực hiện cục bộ hoặc thậm chí được chia sẻ giữa các máy trạm trước khi các thay đổi được chuyển đến máy chủ trung tâm. Sử dụng git, người phát triển có thể liên tục truy xuất các thay đổi từ máy chủ trugn tâm và đưa các thay đổi lên.

GitHub là một dịch vụ kho lưu trữ Git miễn phí cung cấp giao diện đồ họa để tạo và làm việc với các kho lưu trữ, cũng như các tính năng như kiểm soát truy cập, cộng tác và quản lý tác vụ. **GitHub Desktop** cũng có thể được sử dụng để tạo và làm việc với các kho lưu trữ cục bộ và từ xa từ máy tính để bàn. Ví dụ, quy trình đầu tiên trong (initial commit) cho thấy cách tạo kho lưu trữ Git cục bộ bằng GitHub Desktop. Sau khi tạo một kho lưu trữ, bạn sao chép vào bất kỳ tệp dự án nào hiện đang có. Sau đó, nếu bạn thay đổi các tệp này, bạn có thể đưa chúng vào nhánh chính như quy trình thứ hai trong hình này. Khi làm điều đó, các thay đổi cũng thường được đưa lên kho lưu trữ trên GitHub.



Hình 7-2 Giao diện của Github Desktop

Cách tạo kho lưu trữ Git cục bộ với GitHub Desktop :

- Chọn the File > New Repository để hiển thị hộp thoại Create a New Repository. Sau đó, nhập tên và mô tả cho kho lưu trữ.
- Sao chép các tệp dự án của bạn vào kho lưu trữ.

Để biết thêm thông tin về cách sử dụng **GitHub** và **GitHub Desktop**, có thể tham khảo thêm tại địa chỉ: https://github.com và https://desktop.github.com.

7.3 SASS

7.3.1 Giới thiệu

Sass viết tắt của **Syntactally Awesome Style Sheets** cung cấp một cách tiếp cận để xây dựng các định dạng trang Web. Sass giúp cho việc viết CSS theo cách của một ngôn ngữ lập trình, có cấu trúc rõ ràng, rành mạch, dễ phát triển và bảo trì code hơn.

Trước khi có thể cài đặt và sử dụng **Sass**, Ruby phải được cài đặt nó trên máy tính, vì **Sass** dựa trên ngôn ngữ này. Có thể tải xuống và cài đặt **Ruby** phiên bản mới nhất từ Website **https://rubyinstaller.org/downloads**. Sau đó, nhập lệnh gem (**gem install sass)** tại cửa sổ lệnh của Ruby để cài đặt Sass. Mỗi cài đặt này có thể mất vài phút.

Khi Sass đã được cài đặt, bạn có thể biên dịch tệp Sass thành CSS bằng cách sử dụng lệnh sass như sau:

- Trong một thư mục dự án, hãy tạo một thư mục có tên là scss cũng như một thư mục có tên là css.
- Tạo một tệp có phần mở rộng .scss, chẳng hạn như style.scss, trong thư mục scss, sau đó thêm Sass vào tệp .scss.
- Mở cửa sổ lệnh Ruby và nhập lệnh theo định dạng sau:

```
sass scss/filename.scss css/filename.css
```

- Mã CSS đã biên dịch sẽ được lưu trữ trong tệp có phần mở rộng .css.

Tham số tùy chọn cũng có thể được thêm vào:

```
sass --watch sass css
```

Lệnh này sẽ khiến Sass theo dõi tất cả các tệp trong thư mục sass, tự động biên dịch lại bất kỳ tệp nào thay đổi và lưu trữ các tệp CSS mới trong thư mục css. Bằng cách đó, không phải chạy lệnh sass theo cách thủ công trong dấu nhắc lệnh của Ruby mỗi khi các tệp Sass được thay đổi. Tuy nhiên, quy trình biên dịch Sass từ cửa sổ lệnh của Ruby là một quy trình tẻ nhạt. Đó là lý do tại sao Gulp.js thường được sử dụng để theo dõi và biên dịch các tập lệnh Sass, sẽ được trình bày chi tiết ở phần sau của chương này.

7.3.2 Cách sử dụng Sass để lồng các quy tắc kiểu

Ví dụ sau đây thể hiện cách Sass có thể được sử dụng để lồng các quy tắc kiểu. Ở đây, các quy tắc cho các thành phần ul, li và <a> của menu dọc được lồng trong một thành phần nav. Điều này cho thấy cách các kiểu được áp dụng cho các phần tử trong menu. Sau đó, khi Sass được biên dịch, nó sẽ được chuyển thành CSS tiêu chuẩn. Ở đây, có thể thấy rằng Sass dễ hiểu hơn CSS.

Sass cho quy tắc lồng:

```
nav {
    ul {
        list-style-type: none;
        margin-left: 1 cm;
        margin-bottom: 1.5 cm;
    }
    li {
        width: 150px;
        margin-bottom: .5 cm;
        border: 2px solid red;
    }
    a {
        display: block;
        padding: .5 cm, 0 .5cm, 1cm;
        text-decoration: none;
        color: red;
    }
}
```

Css được dịch ra từ Sass:

```
nav ul {
    list-style-type: none;
    margin-left : 1 cm;
    margin-bottom: 1.5 cm; }
nav li {
    width: 150px;
    margin-bottom: .5 cm;
    border: 2px solid red; }
nav a {
    display: block;
    padding: .5 cm, 0 .5cm, 1cm;
    text-decoration: none;
```

```
    color: red; }
```

7.3.3 Sử dụng biến và mixins của Sass

Ví dụ sau đây thể hiện cách sử dụng biến ở trong Sass:

```
$font: Arial, Helvetica, sans-serif;
$color-1: white;
$color-2: navy;
body {
    font-family: $font;
    background-color: $color-1;
}
hl { color: $color-2; }
```

Css được biên dịch từ Sass:

```
body {
    font-family: Arial, Helvetic, sans-serif;
    background-color: white; }
hl {
    color: navy; }
```

Ở đây, tập lệnh khai báo ba biến có tên $font, $color-1 và $color-2 với các giá trị bắt đầu như trong ví dụ. Sau đó là các quy tắc kiểu cho phần tử body và hl sử dụng các biến này. Khi tập lệnh Sass này được biên dịch, các giá trị của biến sẽ thay thế tên biến. Điều này đảm bảo rằng các giá trị giống nhau được sử dụng trên toàn bộ trang Web. Điều này cũng có nghĩa là các giá trị đó trong toàn bộ trang Web có thể thay đổi chỉ bằng cách thay đổi các giá trị trong biến và biên dịch lại tập lệnh Sass.

Ví dụ sử dụng mixin:

```
@mixin box($bgcolor) {
    width: 80px;
    height: 90px;
    background-color: $bgcolor; }

.boxl { @include box(blue); }
.box2 { @include box(yellow); }
```

Css được biên dịch từ Sass:

```
boxl {
width: 80px;
height: 90px;
background-color: blue; }
.box2 {
width: 80px;
height: 90px;
background-color: yellow; }
```

Ở đây, mixin được đặt tên “box” và nó có một tham số có tên là $bgcolor. Mixin này đặt thuộc tính chiều rộng và chiều cao, đồng thời đặt thuộc tính màu nền thành giá trị của biến $bgcolor

được truyền cho nó dưới dạng tham số. Tiếp theo, tập lệnh này định nghĩa hai lớp có tên boxl và box2 sử dụng @include để bao gồm mixin. Trong trường hợp này, @include đầu tiên chuyển giá trị "red" cho biến $bgcolor trong mixin và @include thứ hai chuyển giá trị "blue" cho biến $bgcolor trong mixin.

Khi biên dịch tập lệnh này, nó sẽ tạo CSS được hiển thị trong hình này. Lợi ích của việc sử dụng biến và mixin là có thể thay đổi CSS trong biểu định kiểu chỉ bằng cách thay đổi giá trị của biến và mixin cũng như biên dịch lại. Trong các ví dụ này, điều đó sẽ chỉ thay đổi một vài dòng mã, nhưng trong thực tế, điều đó có thể thay đổi nhiều dòng. Ngoài ra, Sass còn cung cấp các tính năng như toán tử số học, nhập và kế thừa. Đó là lý do tại sao Sass trở nên phổ biến như một ngôn ngữ mở rộng CSS.

7.4 BOOTSTRAP

7.4.1 Giới thiệu

Bootstrap là một framework phát triển Web đầy đủ tính năng giúp xây dựng các trang Web trên máy tính để bàn, thiết bị di động và responsive Website dễ dàng hơn. Trong các chủ đề tiếp theo, các kỹ năng cơ bản để sử dụng Bootstrap sẽ được giới thiệu và thực hành.

Bootstrap bao gồm 4 mô-đun chính được miêu tả trong bảng sau đây:

Bảng 4. Các mô-đun chính của bootstrap (https://getbootstrap.com)

Layout	
Content	
Components	
Utilities	

Mô-đun layout (layout) cung cấp cách bố trí các trang bằng hệ thống lưới 12 cột. Mô-đun nội dung bao gồm một biểu định kiểu có tên bootstrap-reboot.css được xây dựng trên biểu định kiểu normalize.css để cung cấp khả năng hiển thị nhất quán hơn trên các trình duyệt. Mô-đun thành phần cung cấp một số thành phần mà bạn có thể thêm vào trang Web của mình. Và mô-đun tiện ích cung cấp một số tiện ích mà bạn có thể sử dụng để tạo kiểu cho các trang Web của mình.

Ví dụ trong hình dưới đây cho thấy mã cơ bản cần thiết để sử dụng Bootstrap trong một trang Web. Để bắt đầu, bạn viết mã một phần tử liên kết cho biểu định kiểu Bootstrap trong phần tử đầu. Sau đó, ở cuối phần tử nội dung, bạn viết mã một phần tử tập lệnh để bao gồm tệp JavaScript Bootstrap. Phần tử này phải được đặt trước phần tử liên kết cho thư viện jQuery, vì Bootstrap sử dụng jQuery. Ngoài ra, nếu trang sẽ bao gồm bất kỳ thành phần Bootstrap nào sử dụng cửa sổ bật lên, thì bạn cần bao gồm phần tử tập lệnh cho plugin jQuery bật lên trước phần tử tập lệnh cho tệp JavaScript Bootstrap.

```
<!doctype html>
<html lang="en">
```

```
  <head>
    <meta charset="utf-8 11 >
    <meta name="viewport" content="width=device-width, initial-scale=l, shrink-
to-fit=no">
    <link rel="stylesheet" href="css/bootstrap.min.css''>
    <title>A basic Bootstrap starter template</title>
</head>
<body>

<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"></script>
<script src="js/popper.min.js"></script>
<script src="js/bootstrap.min.js"></script>
</body>
</html>
```

Biểu định kiểu Bootstrap đã được tải xuống và đưa vào một thư mục của trang Web có tên là css. Tương tự, các tệp JavaScript Bootstrap và popper đã được tải xuống và đưa vào một thư mục của trang Web có tên js. Tuy nhiên, xin lưu ý rằng có thể bao gồm các tệp này từ CDN. Có thể truy cập trang Web Bootstrap để lấy địa chỉ cho việc này. Lưu ý rằng các phần tử tập lệnh cho các tệp jQuery và JavaScript được mã hóa ở cuối phần tử nội dung, không phải trong phần tử phần đầu. Đây là một thực tế phổ biến hiện nay vì trang được tải vào trình duyệt nhanh hơn vì nó không phải đợi các tệp JavaScript được tải.

7.4.2 Cách sử dụng hệ thống lưới Bootstrap

Bootstrap sử dụng hệ thống lưới dựa trên vùng chứa, hàng và cột. Vùng chứa chứa một hoặc nhiều hàng và một hàng có thể chứa tối đa 12 cột. Bootstrap cung cấp các lớp CSS được xác định trước để hoạt động với hệ thống lưới của nó. Các lớp quan trọng nhất được trình bày trong bảng sau:

container	Chứa hàng hoặc nội dung khác. Căn giữa trong phần tử nội dung, với chiều rộng cụ thể dựa trên kích thước màn hình.
container-fluid	Chứa hàng hoặc nội dung khác. Đặt thành 100% chiều rộng của màn hình.
row	Chứa các cột bên trong một thùng chứa.
col-count	Số lượng cột mà một phần tử sẽ kéo dài.
col-size-count	Số lượng cột mà một phần tử sẽ kéo dài ở một kích thước màn hình nhất định. Các kích thước bao gồm sm (576px trở lên), md (768px trở lên), lg (992px trở lên) và xi (1200px trở lên)

Để bắt đầu, các lớp container và container-fluid được sử dụng để xác định nội dung chính của trang. Sự khác biệt giữa chúng là một phần tử sử dụng lớp vùng chứa được căn giữa trên màn hình và có chiều rộng cụ thể tính bằng pixel dựa trên chiều rộng của khung nhìn. Điều này đôi khi được gọi là layout đóng hộp. Ngược lại, một phần tử sử dụng lớp container-fluid luôn có cùng chiều rộng với khung nhìn. Điều này đôi khi được gọi là fall width layout. Nội dung chính của một trang được chia thành các hàng bằng cách sử dụng lớp row. Sau đó, trong mỗi hàng, bạn có thể mã hóa các thành phần bằng một hoặc nhiều lớp cột kiểm soát cách hiển thị nội dung

trong hàng. Các lớp bạn sử dụng tùy thuộc vào việc bạn muốn tạo trang có layout chiều rộng cố định, layout linh hoạt hay responsive layout.

Để hiểu cách thức hoạt động của tính năng này, các ví dụ tiếp theo sẽ chỉ ra ba cách để bố trí vùng chứa có một hàng và ba phần tử div. Trong ví dụ đầu tiên, lớp vùng chứa được sử dụng để vùng chứa có chiều rộng cố định. Sau đó, mỗi phần tử div trong hàng được gán cho một lớp cho biết số lượng cột mà div kéo dài. Nếu bạn hiển thị vùng chứa này trong trình duyệt, bạn sẽ thấy rằng khi chế độ xem trở nên nhỏ hơn, vùng chứa sẽ co lại với chiều rộng nhỏ hơn khi nó đạt đến điểm dừng cho mỗi kích thước màn hình.

Layout có chiều rộng cố định sử dụng hệ thống lưới Bootstrap:

```
<div class= “container” >
  <div class= “row” >
     <div class= “col-4” >Colwnn 1</div>
     <div class= “col-6” >Colwnn 2</div>
     <div class= “col-2” >Colwnn 3</div>
  </div>
</div>
```

Ví dụ thứ hai cũng tương tự, ngoại trừ vùng chứa sử dụng lớp container-fluid. Do đó, vùng chứa luôn có toàn bộ chiều rộng của khung nhìn và chiều rộng của vùng chứa thay đổi linh hoạt theo chiều rộng của khung nhìn.

Layout linh hoạt với các cột có kích thước bằng nhau

```
<div class= “container” >
  <div class= “row” >
     <div class= “col-4” >Colwnn 1</div>
     <div class= “col-6” >Colwnn 2</div>
     <div class= “col-2” >Colwnn 3</div>
  </div>
</div>
```

Ví dụ thứ ba cho thấy cách triển khai layout đáp ứng với Bootstrap. Ở đây, lớp container-fluid được sử dụng như trong ví dụ thứ hai. Ngoài ra, mỗi phần tử div được gán cho hai lớp cột. Lớp đầu tiên cho biết số lượng cột mà phần tử kéo dài trên một khung nhìn trung bình (768 pixel trở lên) và lớp thứ hai cho biết số lượng cột mà phần tử kéo dài trên một khung nhìn nhỏ (576 pixel trở lên).

Layout có chiều rộng cố định sử dụng hệ thống lưới Bootstrap

```
<div class= “container” >
  <div class= “row” >
     <div class= “col-4” >Colwnn 1</div>
     <div class= “col-6” >Colwnn 2</div>
     <div class= “col-2” >Colwnn 3</div>
  </div>
</div>
```

Lưu ý ở đây rằng nếu vùng chứa được hiển thị trong chế độ xem lớn (992 pixel trở lên) hoặc cực lớn (1200 pixel trở lên), phần tử sẽ trải rộng trên cùng một số cột như trong chế độ xem

trung bình. Đó là bởi vì nếu một lớp không được chỉ định cho kích thước màn hình, số lượng cột của kích thước màn hình nhỏ nhất tiếp theo được chỉ định sẽ được sử dụng. Trên các khung nhìn trung bình hoặc lớn hơn, hai phần tử div đầu tiên, mỗi phần tử kéo dài ba cột và phần tử div thứ ba kéo dài sáu cột. Tuy nhiên, trên một khung nhìn nhỏ, hai phần tử div đầu tiên, mỗi phần tử kéo dài 6 cột và phần tử div thứ ba kéo dài 12 cột. Vì một hàng không thể chứa nhiều hơn 12 cột nên phần tử div thứ ba sẽ được hiển thị bên dưới hai phần tử div đầu tiên.

7.4.3 Cách sử dụng các thành phần Bootstrap

Giống như hệ thống lưới Bootstrap, các thành phần Bootstrap cũng phụ thuộc vào các lớp CSS được xác định trước. Các lớp này có thể sử dụng để định nghĩa các thành phần cũng như để kiểm soát sự xuất hiện và hoạt động của chúng.

Hơn một trăm thành phần (https://getbootstrap.com/) có thể tái sử dụng được xây dựng để cung cấp biểu tượng, danh sách thả xuống, nhóm đầu vào, điều hướng, cảnh báo, ...

Ví dụ với một thành phần là menu drop-down, có thể chuyển đổi, menu theo ngữ cảnh để hiển thị danh sách các liên kết. Thực hiện tương tác với plugin JavaScript dropdown.

Hình 7-3 Ví dụ về một thành phần của Bootstrap

```
<div class="dropdown">
  <button class="btn btn-default dropdown-toggle" type="button"
id="dropdownMenu1" data-toggle="dropdown" aria-haspopup="true" aria-
expanded="true">
    Dropdown
    <span class="caret"></span>
  </button>
  <ul class="dropdown-menu" aria-labelledby="dropdownMenu1">
    <li><a href="#">Action</a></li>
    <li><a href="#">Another action</a></li>
    <li><a href="#">Something else here</a></li>
    <li role="separator" class="divider"></li>
    <li><a href="#">Separated link</a></li>
  </ul>
</div>
```

7.4.4 Biểu tượng (icon)

Cách đơn giản nhất để thêm biểu tượng vào trang HTML là với thư viện biểu tượng, chẳng hạn như Font Awesome. Thêm tên của lớp biểu tượng đã chỉ định vào bất kỳ phần tử HTML nội

tuyến nào (như <i> hoặc). Tất cả các biểu tượng trong các thư viện biểu tượng bên dưới, là các vectơ có thể mở rộng có thể được tùy chỉnh bằng CSS (kích thước, màu sắc, bóng, ...)

Để sử dụng glyphicons Bootstrap, hãy thêm dòng sau vào phần <head> trên trang HTML của:

```
<link rel = "stylesheet" href = "https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7
/css/bootstrap.min.css">
```

Ví dụ

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css
/bootstrap.min.css">
</head>
<body>
<i class="glyphicon glyphicon-cloud"></i>
<i class="glyphicon glyphicon-remove"></i>
<i class="glyphicon glyphicon-user"></i>
<i class="glyphicon glyphicon-envelope"></i>
<i class="glyphicon glyphicon-thumbs-up"></i>
 </body>
</html>
```

7.5 JQUERY

jQuery là một thư viện **JavaScript** . Khẩu hiệu của jQuery là **write less, do more** giúp chúng ta viết mã nguồn JavaScript ít hơn để có thể làm được nhiều việc hơn. Để sử dụng jQuery cần có kiến thức về JavaScript.

Để tích hợp jQuery vào Project:

```
<script src="https://code.jquery.com/jquery-3.6.0.js" integrity="sha256-
H+K7U5CnXl1h5ywQfKtSj8PCmoN9aaq30gDh27Xc0jk="  crossorigin="anonymous"></script>
```

Tạo file html đơn giản như sau:

```
<html>
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, user-scalable=no, initial-
scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
   <meta http-equiv="X-UA-Compatible" content="ie=edge">
   <title>Document</title>
```

```
</head>
<body>
<h1>Hello</h1>
<script src="https://code.jquery.com/jquery-3.6.0.js" integrity="sha256-
H+K7U5CnXl1h5ywQfKtSj8PCmoN9aaq30gDh27Xc0jk="  crossorigin="anonymous"></script>

<script>
   $( "h1" ).remove();
</script>

</body>
</html>
```

Chạy thử sẽ thấy thẻ h1 ban đầu xuất hiện với chữ Hello, sau đó bị mất đi bởi jQuery. Với jQuery, để thực hiện một phương thức trên một đối tượng ta sẽ viết như ví dụ trên:

```
$("h1").remove();
```

Phần **<h1>** được gọi là selectors, còn remove() là phương thức. Hầu hết các mã lệnh jQuery được viết theo cách này. Ví dụ:

```
$( "#title" ).html( "Hello world" );
```

Tuy nhiên jQuery cũng cung cấp thêm các hàm hỗ trợ, ví dụ:

```
$.trim( "    lots of extra whitespace    " );
```

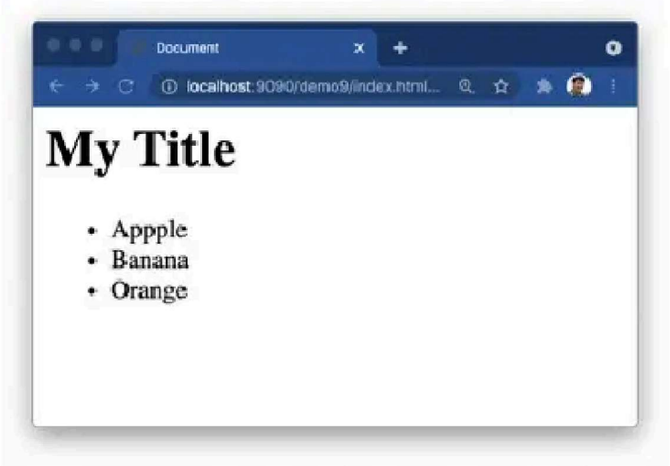
Như vậy trong jQuery sẽ có $(). và $. Khi lập trình sẽ kết hợp cả việc sử dụng $() và $, ví dụ như sau:

```
<h1 id="title"></h1>
<script>
   $("#title").append($.trim( "    My Title    " ))
</script>
```

Hoặc:

```
<ul id="list"></ul>

<script>
   $.each(["Appple", "Banana", "Orange"], function (idx, val) {
       $("#list").append("<li>" + val + "</li>")
   });
</script>
```

Kết quả sau khi thực hiện các câu lệnh trên:



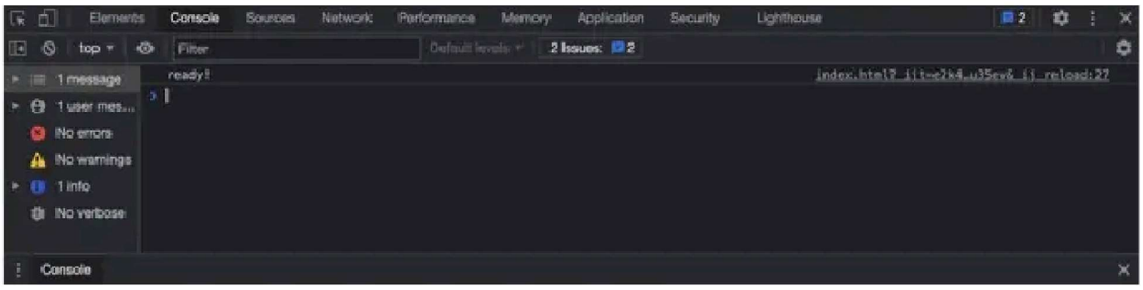
Hình 7-4 Danh sách in ra bởi jQuery

7.5.1 $().ready()

Một Website không thể thao tác được khi chưa ở trong trạng thái sẵn sàng (ready). jQuery có thể phát hiện trạng thái sẵn sàng và thực thi các câu lệnh khi mọi thứ là sẵn sàng bằng cách sử dụng hàm $().ready():

```
$().ready(function() {
   console.log( "ready!" );
});
```

Kết quả sau khi chạy sẽ hiển thị tại cửa sổ Developer tools.



7.5.2 Các bộ chọn

jQuery hỗ trợ hầu hết các CSS3 selectors. Ví dụ chọn theo ID:

```
$("#myId" );
```

Chọn theo tên lớp:

```
$(".myClass" );
```

Chọn theo tên và giá trị của thuộc tính:

```
$("input[name='first_name']" );
```

Sử dụng CSS kết hợp:

```
$("#contents ul.people li" );
```

Chọn phần tử giả (pseudo):

```
$("a.external:first" );
$("tr:odd" );
$("#myForm :input" );
$("div:visible" );
$("div:gt(2)" );
$("div:animated" );
```

Sau khi lựa chọn được các thành phần, có thể lựa chọn tiếp (lọc) để được các phần tửtheo đúng ý, ví dụ:

```
$( "div.foo" ).has( "p" );         // div.foo elements that contain <p> tags
$( "h1" ).not( ".bar" );           // h1 elements that don't have a class of bar
$( "ul li" ).filter( ".current" ); // unordered list items with class of current
$( "ul li" ).first();              // just the first unordered list item
$( "ul li" ).eq( 5 );              // the sixth
```

Khi đã có các Element, có thể lưu vào biến và sử dụng biến đó:

```
var divs = $( "div" );
```

7.5.3 Lấy và thiết lập giá trị

Các phần tử sau khi được chọn có thể sử dụng nhiều các phương thức. Một số phương thức thông dụng:

- .html() – Lấy ra toàn bộ mã HTML bên trong của phần tử.
- .text() – Lấy ra văn bản bên trong phần tử HTML.
- .attr() – Lấy ra giá trị thuộc tính.
- .width() – Lấy hoặc thiết lập độ rộng.
- .height() – Lấy hoặc thiết lập độ cao.
- .val() – Lấy ra giá trị của phần tử Form.

Các phương thức trên dùng để đọc hoặc thiết lập các giá trị thuộc tính cho các Element. Ví dụ:

```
$("h1").html(); //Đọc giá trị
$("h1").html("Xin chào");  //Thiết lập giá trị
```

Các phương thức sau dùng để di chuyển phần tử:

- .insertAfter() - Chèn vào sau
- .insertBefore() - Chèn vào trước
- .appendTo() - Ghép vào cuối
- .prependTo() - Ghép vào đầu

Ví dụ HTML ban đầu:

```
<h1 id="title">
   Title
</h1>
Content
<ul id="list">
   <li>Apple</li>
   <li>Banana</li>
   <li>Orange</li>
</ul>
```

Kết quả sau khi **$("#list").insertAfter("#title");**

```
<h1 id="title">
   Title
</h1>
<ul id="list">
   <li>Apple</li>
   <li>Banana</li>
   <li>Orange</li>
</ul>
Content
```

Kết quả sau khi **$("#list").insertBefore("#title");**

```
<ul id="list">
   <li>Apple</li>
   <li>Banana</li>
   <li>Orange</li>
</ul>
<h1 id="title">
   Title
</h1>
Content
```

Kết quả sau khi **$("#list").appendTo("#title");**

```
<h1 id="title">
   Title
   <ul id="list">
      <li>Apple</li>
      <li>Banana</li>
      <li>Orange</li>
   </ul>
</h1>
Content
```

Kết quả sau khi **$("#list").prependTo($("#title"));**

```
<h1 id="title">
   <ul id="list">
      <li>Apple</li>
      <li>Banana</li>
      <li>Orange</li>
   </ul>
   Title
</h1>
```

```
Content
```

Để copy dùng hàm clone(). Ví dụ kết quả sai khi **$("#list").clone().insertAfter("#title")**

```
<h1 id="title">
   Title
</h1>
<ul id="list">
   <li>Apple</li>
   <li>Banana</li>
   <li>Orange</li>
</ul>
Content
<ul id="list">
   <li>Apple</li>
   <li>Banana</li>
   <li>Orange</li>
</ul>
```

Để xóa 1 element có thể dùng phương thức remove(). Kết quả sau khi **$("#list").remove()**

```
<h1 id="title">
   Title
</h1>
Content
```

Nếu sử dụng phương thức empty() thì chỉ xóa nội dung bên trong. Kết quả sau khi **$("#list").empty()**

```
<h1 id="title">
   Title
</h1>
<ul id="list">
</ul>
Content
```

Sử dụng phương thức **.attr()** để lấy hoặc thay đổi các thuộc tính.

```
$( "#myDiv a:first" ).attr( "href", "newDestination.html" );
```

7.5.4 Tìm kiếm phần tử xung quanh

Ví dụ với HTML ban đầu:

```
<div class="body">
   <h1 id="title">Title</h1>
   Content
   <ul id="list">
       <li>Apple</li>
       <li>Banana</li>
       <li>Orange</li>
   </ul>
</div>
```

Hàm parent sẽ về cha của 1 element. Ví dụ kết quả khi thực hiện **$("#title").parent().addClass("bg-dark")**

```
<div class="body bg-dark">
   <h1 id="title">Title</h1>
   Content
   <ul id="list">
       <li>Apple</li>
       <li>Banana</li>
       <li>Orange</li>
   </ul>
</div>
```

Cũng có thể sử dụng **$("#title").parents()** để xác định tất cả các parent của 1 Element, hoặc sử dụng **$("#title").closest("div")** để tìm parent có thẻ div gần nhất.

Hàm children() để tìm ra các children. Ví dụ kết quả khi thực hiện **$(".body").children().addClass("bg-dark"):**

```
<div class="body">
   <h1 id="title" class="bg-dark">Title</h1>
   Content
   <ul id="list" class="bg-dark">
       <li>Apple</li>
       <li>Banana</li>
       <li>Orange</li>
   </ul>
</div>
```

Nếu chỉ muốn tìm thẻ **h1** bên trong: **$(".body").children("h1").** Còn nếu muốn tìm ở cấp trong nữa thì dùng hàm **find()**, ví dụ: **$(".body").find("li")**.

Để tìm các Element cùng cấp, ta sử dụng các hàm next(), prev(), nextAll(), preveAll(),...

Ví dụ kết quả khi chạy **$("#title").next().addClass("bg-dark")**.

```
<div class="body">
   <h1 id="title">Title</h1>
   Content
   <ul id="list" class="bg-dark">
       <li>Apple</li>
       <li>Banana</li>
       <li>Orange</li>
   </ul>
</div>
```

7.5.5 Định dạng kiểu

Hàm css() để lấy ra thuộc tính css của 1 Element, ví dụ, để lấy thuộc tính **fontSize**:

```
console.log($("h1").css("fontSize"))
```

Hàm css cũng dùng để thiết lập thuộc tính:

```
$("h1").css("fontSize", "100px")
```

Thiết lập nhiều thuộc tính:

```
$("h1").css({
```

```
   fontSize: "100px",
   color: "red"
});
```

Các hàm **addClass**(), **removeClass**(), hoặc **toggleClass**() được sử dụng để **thêm**, **xóa**, hoặc **tắt/bật** class. Ví dụ để thêm 1 class:

```
h1.addClass( "big" );
```

Các phần tử cũng có thể chứa dữ liệu, để lưu và lấy ra dữ liệu, sử dụng hàm data. Ví dụ đơn giản:

```
$("#title").data("x", 12);
```

7.5.6 Sự kiện

jQuery cho phép thiết lập sự kiện cho các Element một cách dễ dàng. Ví dụ:

```
$("#title").click(function (){
   console.log( "You clicked the title!" );
})
```

Sự kiện click cũng có thể gán bằng hàm on:

```
$("#title").on("click",function (){
   console.log( "You clicked the title!" );
})
```

Các sự kiện khác cũng có thể được gán, và có thể lấy ra các giá trị, liên quan đến sự kiện

```
$( "input" ).on(
   "change",
   function( event ) {
       console.log(event.target.value);
    }
);
```

Để gán 1 sự kiện nhưng chỉ cho phép dùng 1 lần:

```
$("#title").one("click", function () {
   console.log("You clicked the title!");
})
```

Để xóa 1 event, dùng hàm off():

```
$("#title").off("click")
```

Muốn một sự kiện thực thi (nhưng không cần có sự tác động):

```
$("#title").trigger("click");
```

7.5.7 Hiệu ứng

jQuery có thể ẩn hiện thông tin bằng hàm show() và hide(). Ví dụ:

```
$( "p" ).hide();
// Instantaneously show all divs that have the hidden style class
$( "div.hidden" ).show();
```

Việc ẩn có thể thực hiện với tốc độ chậm bằng cách thêm các tùy chọn 'slow" "normal" và "fast". Ví dụ:

```
// Slowly hide all paragraphs
$( "p" ).hide( "slow" );
// Quickly show all divs that have the hidden style class
$( "div.hidden" ).show( "fast" );
```

Thời gian ẩn hiện cũng có thể được thiết lập thông qua số milli giây:

```
// Hide all paragraphs over half a second
$( "p" ).hide( 500 );
// Show all divs that have the hidden style class over 1.25 seconds
$( "div.hidden" ).show( 1250 );
```

Các hàm slideUp(), và slideDown(), cũng có thể sử dụng để ẩn hiện các đối tượng:

```
// Hide all paragraphs using a slide up animation over 0.8 seconds
$( "p" ).slideUp( 800 );
// Show all hidden divs using a slide down animation over 0.6 seconds
$( "div.hidden" ).slideDown( 600 );
```

Tạo ra hiệu ứng Fade bằng cách gọi các hàm fadeOut() và fadeIn()

```
// Hide all paragraphs using a fade out animation over 1.5 seconds
$( "p" ).fadeOut( 1500 );
// Show all hidden divs using a fade in animation over 0.75 seconds
$( "div.hidden" ).fadeIn( 750 );
```

Có thể chuyển trạng thái ẩn hiện bằng hàm toogle()

```
// Instantaneously toggle the display of all paragraphs
$( "p" ).toggle();
// Slowly toggle the display of all images
$( "img" ).toggle( "slow" );
// Toggle the display of all divs over 1.8 seconds
$( "div" ).toggle( 1800 );
```

Để thực hiện một công việc nào đó sau khi chuyển động kết thúc, chỉ cần viết thêm hàm đó, ví dụ:

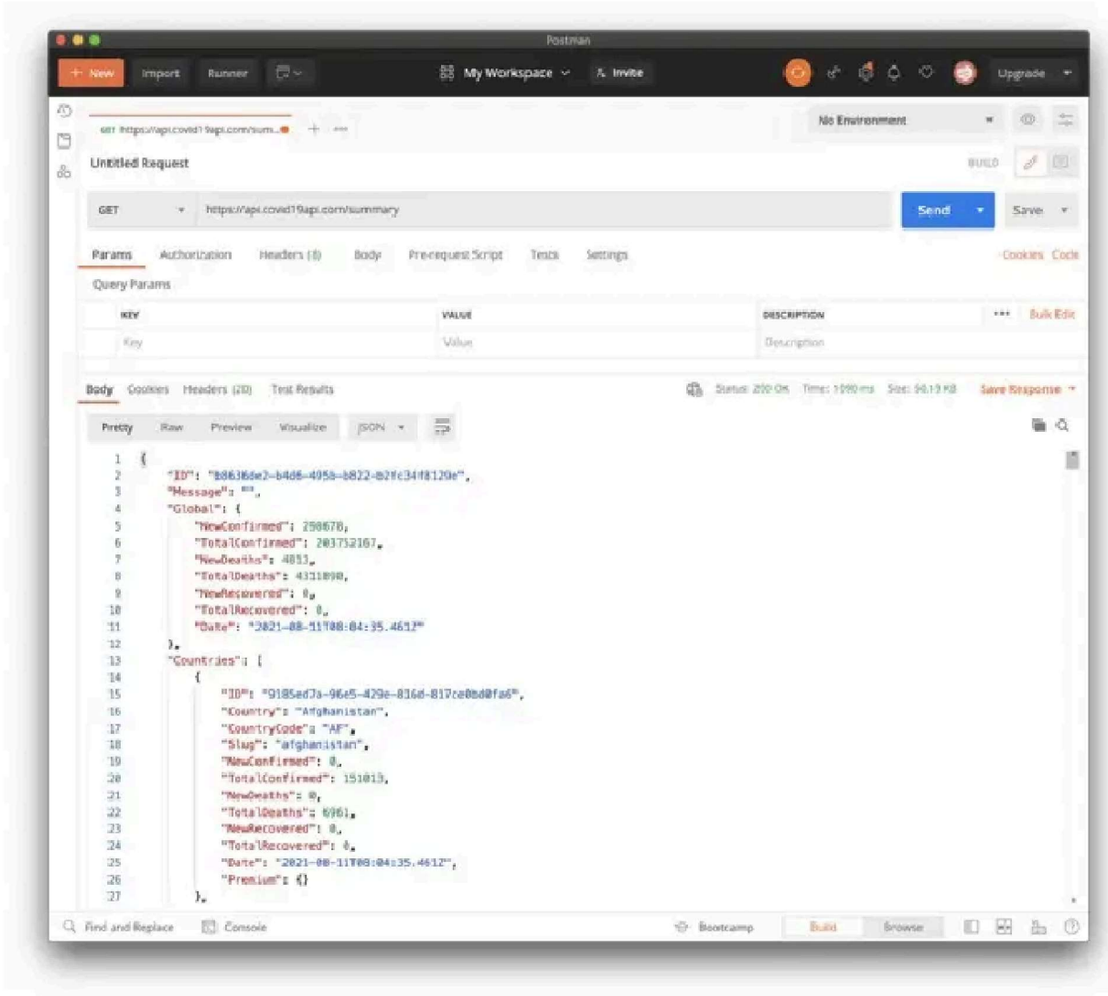
```
// Fade in all hidden paragraphs; then add a style class to them (not quite
right)
$( "p.hidden" ).fadeIn( 750 ).addClass( "lookAtMe" );
```

Có thể khai báo một hâm callback như sau, và hàm sẽ được gọi sau khi effect kết thúc:

```
// Fade in all hidden paragraphs; then add a style class to them (correct with
animation callback)
$( "p.hidden" ).fadeIn( 750, function() {
   // this = DOM element which has just finished being animated
   $( this ).addClass( "lookAtMe" );
});
```

7.5.8 Sử dụng Ajax

jQuery cũng hỗ trợ AJAX, giúp gửi yêu cầu đến Server và xử lý phi động bộ. Giả sử Server phục vụ các Request và trả về 1 kết quả, ví dụ:



Ví dụ để gọi một giao thức **get** khi bấm vào title:

```
$("#title").click(function () {
   $.get("https://api.covid19api.com/summary", function (data)
       $("#title").html(data.Global.NewConfirmed)
   });
});
```

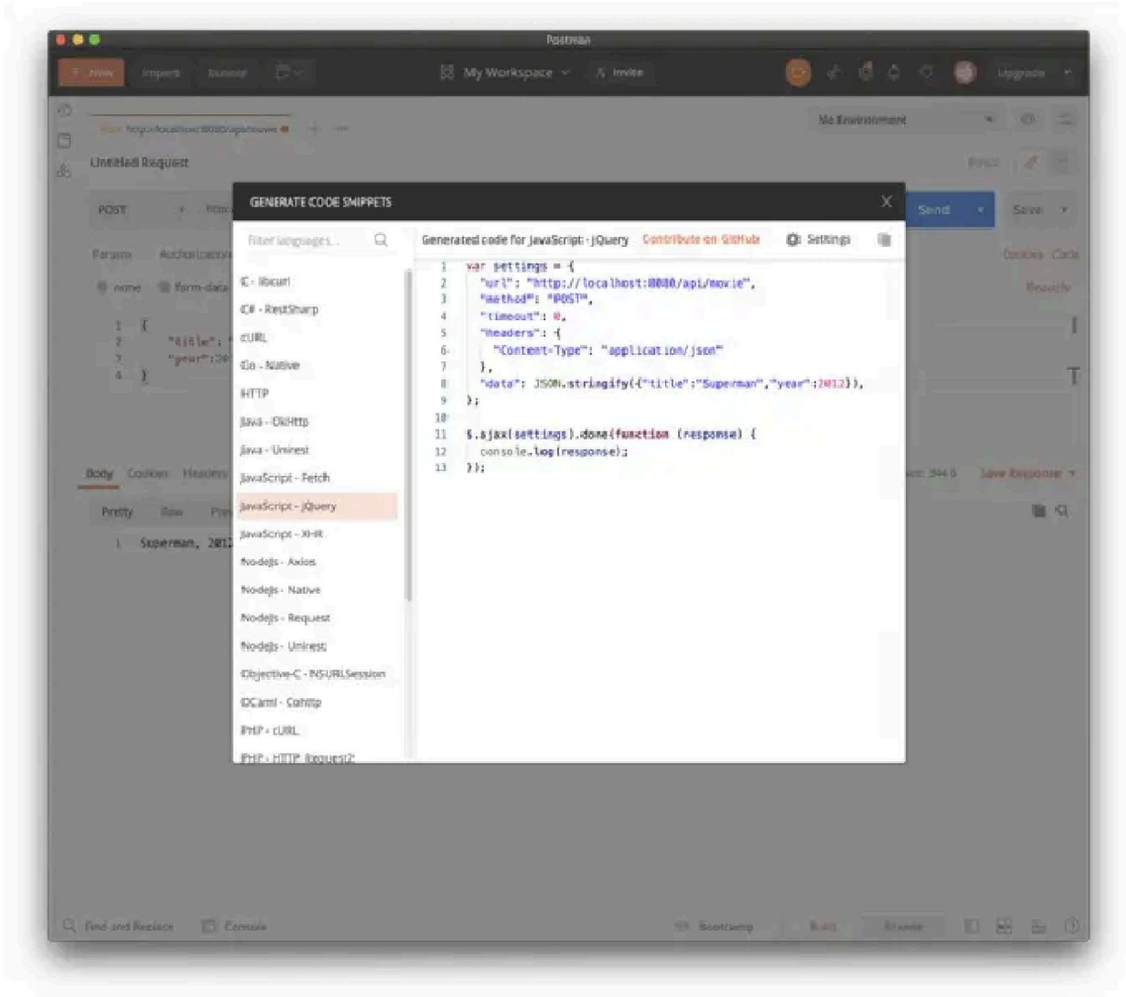
Ngoài hàm get() cũng có thể sử dụng các hàm post(). Với các phương thức khác, có thể sử dụng hàm ajax() như sau:

```
$.ajax({
   url: "https://api.covid19api.com/summary",
   type: "GET"
}).done(function (data) {
   $("#title").html(data.Global.NewConfirmed)
}).fail(function (xhr, status, errorThrown) {
   $("#title").html("Sorry, there was a problem!")
}).always(function (xhr, status) {
   console.log("The request is complete!")
});
```

Data có thể được gửi cùng 1 giao thức post:

```
$.ajax({
   "url": "http://localhost:8080/api/movie",
   "method": "POST",
   "headers": {
       "Content-Type": "application/json"
   },
   "data": JSON.stringify({"title":"Superman","year":2012}),
}).done(function (response) {
   console.log(response);
});
```

Để nhanh ta nên dùng Postman, gửi request sau đó xuất ra mã Jquery Ajax



Khi submit 1 form, ta nên kiểm tra điều kiện, sau đó gửi request bằng Ajax, và sử dụng **event.preventDefault()**; để form ko gửi request theo cách thông thường.

```
<form action="http://localhost:8080/api/movie" method="post" id="form">
    <input type="text" name="title" id="title" class="required">
    <input type="text" name="year" id="year" class="required">
    <input type="submit">
</form>

<script>
    $().ready(function () {
        $("#form").submit(function (event) {
            // If .required's value's length is zero
            if ($(".required").val().length === 0) {
                //Show message here
            } else {
                var settings = {
                    "url": "http://localhost:8080/api/movie",
                    "method": "POST",
                    "timeout": 0,
                    "headers": {
```

```
                    "Content-Type": "application/x-www-form-urlencoded"
                },
                "data": {
                    "title": $("#title").val(),
                    "year": $("#year").val()
                }
            };
            $.ajax(settings).done(function (response) {
                $("#form").append("<br>" + response)
            });
        }
        event.preventDefault();
    });
  })
</script>
```

7.5.9 Các chương trình cài cắm

jQuery có rất nhiều các chương trình cài cắm (Plugin), được đưa lên https://plugins.jquery.com/, ngoài ra cũng có nhiều các thư viện hoạt động dựa trên jQuery như:

- https://datatables.net/
- https://muuri.dev/
- https://facedetection.jaysalvat.com/

THỰC HÀNH

Để sử dụng datatables vào trang https://datatables.net/

Sử dụng CSS và JS:

```
<link rel="stylesheet" type="text/css"
href="https://cdn.datatables.net/1.11.2/css/jquery.dataTables.css">
<script type="text/JavaScript " charset="utf8"
src="https://cdn.datatables.net/1.11.2/js/jquery.dataTables.js"></script>
```

Tạo trang HTML:

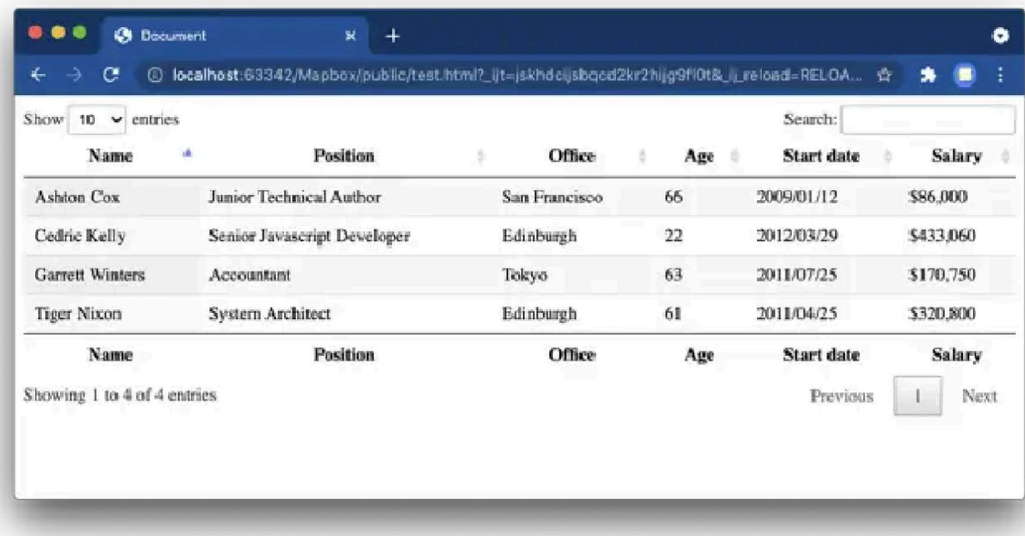
```
<table id="example" class="display" style="width:100%">
        <thead>
            <tr>
                <th>Name</th>
                <th>Position</th>
                <th>Office</th>
                <th>Age</th>
                <th>Start date</th>
                <th>Salary</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>Tiger Nixon</td>
                <td>System Architect</td>
                <td>Edinburgh</td>
```

```
                <td>61</td>
                <td>2011/04/25</td>
                <td>$320,800</td>
            </tr>
            <tr>
                <td>Garrett Winters</td>
                <td>Accountant</td>
                <td>Tokyo</td>
                <td>63</td>
                <td>2011/07/25</td>
                <td>$170,750</td>
            </tr>
            <tr>
                <td>Ashton Cox</td>
                <td>Junior Technical Author</td>
                <td>San Francisco</td>
                <td>66</td>
                <td>2009/01/12</td>
                <td>$86,000</td>
            </tr>
            <tr>
                <td>Cedric Kelly</td>
                <td>Senior JavaScript  Developer</td>
                <td>Edinburgh</td>
                <td>22</td>
                <td>2012/03/29</td>
                <td>$433,060</td>
            </tr>
        </tbody>
        <tfoot>
            <tr>
                <th>Name</th>
                <th>Position</th>
                <th>Office</th>
                <th>Age</th>
                <th>Start date</th>
                <th>Salary</th>
            </tr>
        </tfoot>
    </table>
```

Mã JS:

```
$(document).ready(function() {
    $('#example').DataTable();
} );
```

Kết quả:



Để tải dữ liệu từ nguồn online, ví dụ: https://datatables.net/examples/ajax/data/arrays.txt, sử dụng Ajax được tích hợp sẵn.

Sửa lại table như sau:

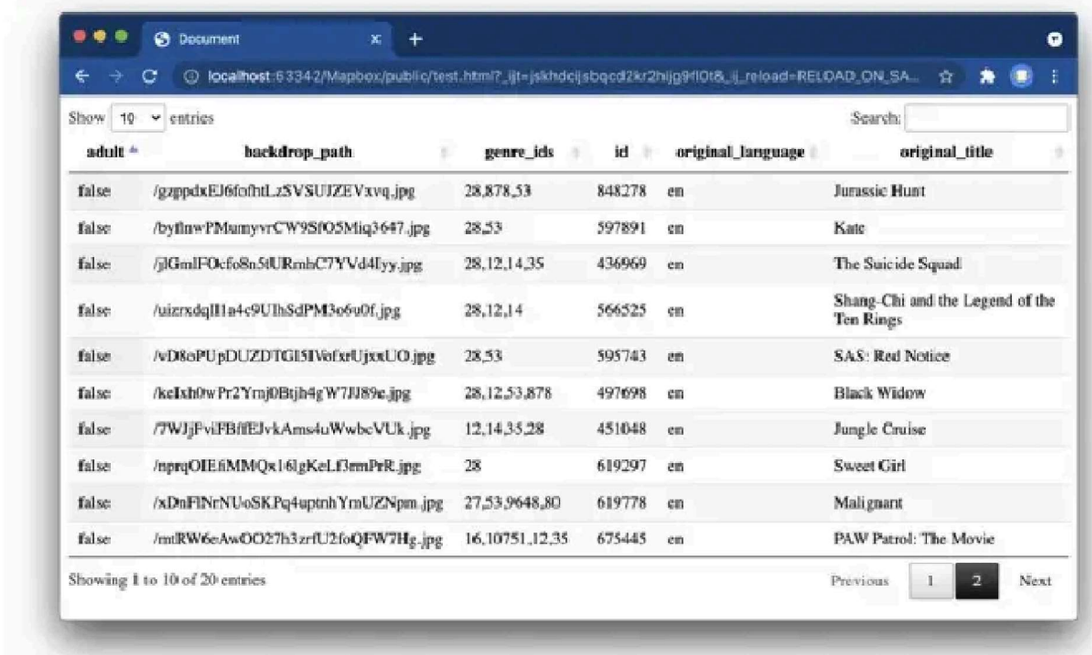
```
<table id="example" class="display" style="width:100%">
        <thead>
            <tr>
                <th>Name</th>
                <th>Position</th>
                <th>Office</th>
                <th>Extn.</th>
                <th>Start date</th>
                <th>Salary</th>
            </tr>
        </thead>
        <tfoot>
            <tr>
                <th>Name</th>
                <th>Position</th>
                <th>Office</th>
                <th>Extn.</th>
                <th>Start date</th>
                <th>Salary</th>
            </tr>
        </tfoot>
</table>
```

Sửa lại mã JS:

```
$(document).ready(function() {
   $('#example').DataTable( {
       ajax: {
```

```
            url:
'http://api.themoviedb.org/3/discover/movie?api_key=62da0532e34e31705aee0abf47dd
eeff',
            dataSrc: 'results'
        },
        "columns": [
            { "data": "adult" },
            { "data": "backdrop_path" },
            { "data": "genre_ids" },
            { "data": "id" },
            { "data": "original_language" },
            { "data": "original_title" },
        ]
    } );
});
```

Kết quả:



CÂU HỎI ÔN TẬP LÝ THUYẾT

1. Emmet có sẵn cho các trình soạn thảo mã nguồn nào?

A. Sublime Text, Photoshop và Sketch

B. Visual Studio, Illustrator và Figma

C. Sublime Text, Visual Studio Code và Atom

D. Tất cả các phương án đều đúng

2. Lệnh nào dùng để khởi tạo một kho lưu trữ mới trên Git?

A. git clone
B. git init
C. git add
D. git commit

3. Lệnh nào dùng để kiểm tra trạng thái của các tệp trong kho lưu trữ Git?

A. git add
B. git status
C. git commit
D. git push

4. Lệnh nào dùng để thêm tệp hoặc thư mục mới vào kho lưu trữ Git?

A. git add
B. git clone
C. git push
D. git pull

5. Lệnh nào dùng để kết hợp hai nhánh lại với nhau trên kho lưu trữ Git?

A. git branch
B. git checkout
C. git merge
D. git pull

6. Lệnh nào được sử dụng để tạo một bảng với Bootstrap?

A. <table class="table">
B.
C. <img src="image.jpg" alt="Image">
D. <button class="btn btn-primary">Button</button>

7. Lệnh nào dùng để chờ tài liệu HTML được tải xong trước khi thực hiện các tác vụ trong jQuery?

A. $(document).ready()
B. $(window).load()
C. $(document).load()
D. $(window).ready()

8. Lệnh nào dùng để chọn tất cả các phần tử div trong HTML bằng jQuery?

A. $("div")
B. $("#div")
C. $(".div")
D. "$div"

9. Lệnh nào dùng để lấy giá trị của thuộc tính trong một phần tử HTML trong jQuery?

A. .attr()
B. .val()
C. .text()
D. .html()

10. Lệnh nào dùng để thay đổi nội dung của một phần tử HTML trong jQuery?

A. .text()
B. .html()
C. .val()
D. .append()

11. Sass là viết tắt của từ gì?

A. Scalable Application Style Sheets
B. Syntactically Awesome Style Sheets
C. Simple Accessible Style Sheets

D. Structured Application Style Sheets

12. Để khai báo một mixin trong Sass, dùng từ khóa nào?

A. @import

B. @include

C. @extend

D. @mixin

13. Để thực hiện đảo ngược một chuỗi trong Sass, dùng hàm nào?

A. invert()

B. reverse()

C. flip()

D. Tất cả đều sai

BÀI TẬP TỰ THỰC HÀNH

Hãy truy cập trang https://github.com/ và thực hiện:

- Tạo tài khoản ở Website trên để lưu trữ các repository và quản lý mã nguồn.
- Đưa những mã nguồn đã làm ở các phần trước lên các repository online.
- Triển khai thêm Bootstrap và jQuery vào các bài tập tự thực hành ở chương trước đó (như website giới thiệu sản phẩm ở chương 6).

TÀI LIỆU THAM KHẢO

[1] JavaScript and jQuery: Interactive Front-End Web Development (2014), Jon Duckett, Wiley

[2] Jump Start Sass: Get Up to Speed With Sass in a Weekend (2016), Hugo Giraudel, SitePoint

[3] Build a Website Now: A Beginner's Guide to Web Development: HTML, CSS and Bootstrap (2019), Riwanto Megosinarso, Independently published

CHƯƠNG 8: TRIỂN KHAI WEBSITE

Chương này hướng dẫn cách thức đưa một trang Web từ máy tính cục bộ lên môi trường mạng Internet. Để làm điều đó, người học sẽ làm quen với các khái niệm như tên miền (domain), nơi lưu trữ trang Web trên không gian mạng Internet (Web host), ứng dụng giúp chuyển các file mã nguồn trang Web từ máy tính cục bộ đến một máy chủ Web có kết nối tới mạng Internet (FTP client). Bên cạnh đó, người học cũng được trang bị một số kinh nghiệm trong việc kiểm thử, tối ưu hoá việc vận hành Website trên môi trường trực tuyến.

8.1 LỰA CHỌN TÊN MIỀN (DOMAIN)

8.1.1 Tên miền (domain)

Tên miền (domain) là địa chỉ của một Website hoạt động trên môi trường mạng Internet. Mục đích việc sử dụng tên miền là để cung cấp một hình thức đại diện tức dùng tên gọi dễ nhận biết thay cho địa chỉ của máy chủ lưu trữ Website (địa chỉ IP). Với việc sử dụng địa chỉ "dạng chữ cái" không trùng nhau thay cho địa chỉ dạng "dãy số", tên miền cho phép người dùng Internet dễ dàng tìm kiếm và truy cập các Website. Tính uyển chuyển của hệ thống tên miền cho phép nhiều địa chỉ IP có thể được gán vào một tên miền, hoặc nhiều tên miền cùng trỏ đến một địa chỉ IP (một máy chủ gắn với một địa chỉ IP có thể cung cấp nơi lưu trữ cho nhiều Website khác nhau, cho nhiều tên miền khác nhau).

Ví dụ 1: người dùng có thể sử dụng tên miền neu.edu.vn thay vì địa chỉ IP: 125.212.138.22.

Ví dụ 2: máy chủ tại địa chỉ IP 113.161.108.140 vận hành một số Website sau:

- kilo.vn
- cokhikhaviet.com.vn
- drgauges.net
- pretem.com

Tên miền được tạo thành từ các nhãn khác rỗng phân cách nhau bằng dấu chấm (.). Cụ thể:

- Nhãn giới hạn bởi các **chữ cái** từ a đến z (không phân biệt viết hoa/thường), **chữ số** từ 0 đến 9, và **dấu gạch ngang** (-).
- Dấu gạch ngang (-) không được xuất hiện ở **đầu** hoặc **cuối** của nhãn.
- Chiều dài của nhãn < 63 ký tự.
- Tổng chiều dài tên miền không được vượt quá 255 ký tự.

Mọi tên miền đều kết thúc bằng một **tên miền cấp cao nhất** (TLD - Top Level Domain), nó thường chứa một trong những **tên miền cấp cao nhất dùng chung** như .com, .net, .org... hoặc một **tên miền quốc gia cấp cao nhất** như .vn, .uk, .us... Việc tạo lập và ủy quyền tên miền quốc gia được thực hiện bởi Tổ chức cấp phát số hiệu Internet (**IANA** - *Internet Assigned Numbers Authority*).

8.1.2 Lựa chọn nơi mua tên miền

Người sở hữu tên miền thường được gọi là chủ tên miền, được độc quyền quản lý và sử dụng tên miền đó. Để sở hữu tên miền, hiện có khá nhiều công ty cung cấp dịch vụ bán tên miền. Ở Việt Nam, có thể lựa chọn một số công ty uy tín thực hiện cung cấp dịch vụ này như:

- Công ty Phần mềm Nhân Hoà: **https://nhanhoa.com**
- Công ty Cổ phần giải pháp mạng Bạch Kim: **https://www.bkns.vn**
- Công ty cổ phần Mắt Bão: **https://www.matbao.com**
- Công ty TNHH Phần mềm iNET: **https://inet.vn**

Bên cạnh đó, cũng có thể lựa chọn một số tổ chức cung cấp tên miền miễn phí như:

- **https://www.freenom.com**
- **https://www.hostinger.com**

8.2 LỰA CHỌN NƠI ĐẶT WEBSITE (WEB HOST)

8.2.1 Dịch vụ lưu trữ Website (Web host)

Dịch vụ lưu trữ Web (Web host) là dịch vụ cung cấp không gian lưu trữ Website trên máy chủ. Nhà cung cấp sẽ tạo ra các "máy chủ Web" nhằm giúp khách hàng có không gian lưu trữ để vận hành các Website. Có nhiều loại dịch vụ lưu trữ Web có thể kể đến như:

- **Hosting**: là dịch vụ lưu trữ nhiều trang Web trên một máy chủ có kết nối tới mạng Internet. Mỗi trang Web có phân vùng riêng. Dịch vụ này là một lựa chọn kinh tế cho người dùng thông qua việc chia sẻ tổng chi phí thuê, quản lý, vận hành, bảo trì máy chủ.
- **Windows hosting**: là dịch vụ lưu trữ, các trang Web được vận hành trên một máy chủ chạy hệ điều hành Windows Server kết nối mạng Internet.
- **Linux hosting**: là một dịch vụ lưu trữ, các trang Web được vận hành trên một máy chủ chạy hệ điều hành Linux Server kết nối mạng Internet.
- **Free Web hosting**: là dịch vụ lưu trữ trang Web miễn phí, thường có quảng cáo và một số hạn chế khác. Free Web Hosting sẽ cung cấp một tên miền phụ (yoursite.example.com) hoặc một thư mục (www.example.com/yoursite) hoặc có thể cho phép sử dụng tên miền của khách hàng với một vài điều kiện kèm theo.
- **Reseller hosting**: là dịch vụ lưu trữ của máy chủ Web mà khách hàng có khả năng được phép phân bổ lại tài nguyên của họ như ổ cứng, băng thông... tới các tổ chức/cá nhân khác.
- **VPS Hosting**: là một máy chủ riêng ảo (VPS - Virtual Private Server). VPS là một phương pháp phân vùng một máy chủ vật lý thành nhiều máy chủ ảo. Mỗi máy chủ ảo có thể chạy hệ điều hành riêng và được khởi động lại mà không ảnh hưởng đến các máy chủ khác.
- **Email hosting**: là dịch vụ thư điện tử nhằm chứng thực Email khách hàng gửi đi gắn với tên miền của họ. Khách hàng được phép tuỳ chỉnh cấu hình, số lượng tài khoản,

dung lượng lưu trữ và được phép sử dụng các dịch vụ phổ biến liên quan đến thư điện tử.

- **File hosting**: là dịch vụ lưu trữ tệp tin trực tuyến trên máy chủ Web. Thông thường, nhà cung cấp dịch vụ sẽ cho phép khách hàng khả năng truy cập qua giao thức FTP hoặc các công cụ chuyên biệt để tải, quản lý, lưu trữ và chia sẻ các tệp tin lên máy chủ Web.

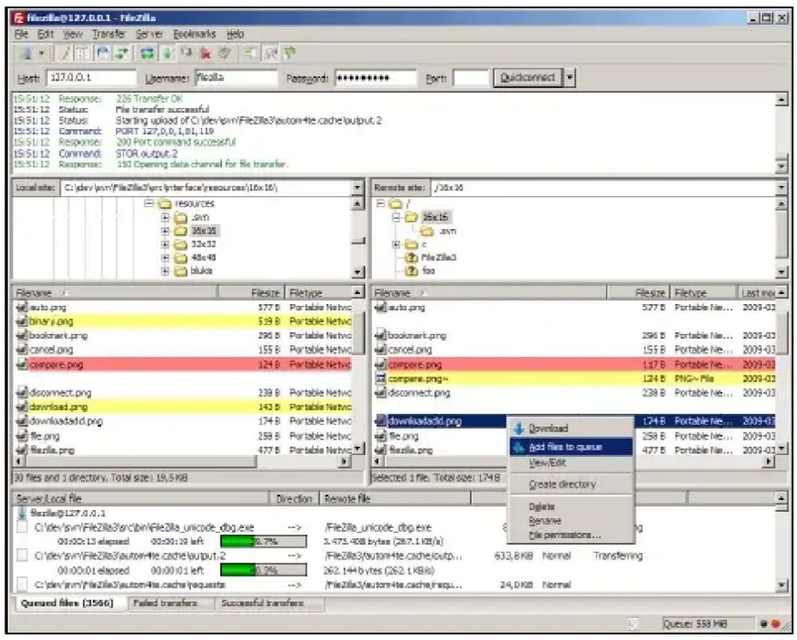
8.2.2 Lựa chọn nơi mua Web host

Để có thể mua được Web host, hiện có khá nhiều công ty cung cấp dịch vụ Web host. Ở Việt Nam, có thể lựa chọn một số công ty uy tín thực hiện cung cấp dịch vụ này (như Nhân Hoà, BKNS, Mắt Bão, iNET,…) để mua Web host. Khi quyết định mua Web host, cần cân đối một số thông tin như: môi trường vận hành của Web host (hệ điều hành, hệ quản trị cơ sở dữ liệu,…), dung lượng lưu trữ, kinh phí duy trì/tháng, thời gian sử dụng dự kiến,… để từ đó lựa chọn phương án phù hợp nhất.

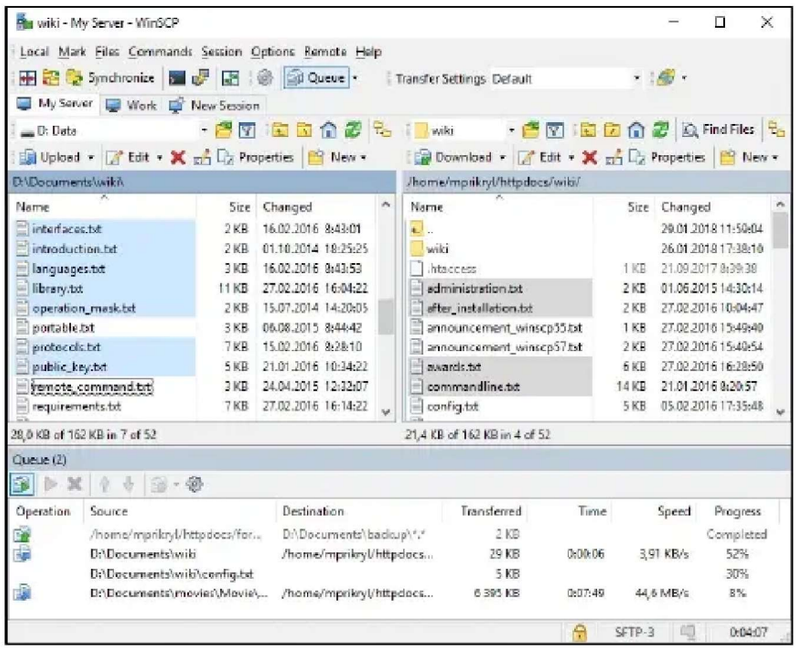
8.2.3 Chuyển Website lên Web host

Để chuyển toàn bộ mã nguồn trang Web tới Web host. Nếu Web host đó cung cấp dịch vụ FTP *(File Transfer Protocol)*, có thể sử dụng ứng dụng FTP Client, nhằm chuyển toàn bộ mã nguồn từ máy cá nhân tới Web host một cách đơn giản. Dưới đây là một số ứng dụng FTP Client thường được sử dụng trong thực tế:

FileZilla Client:



WinSCP:



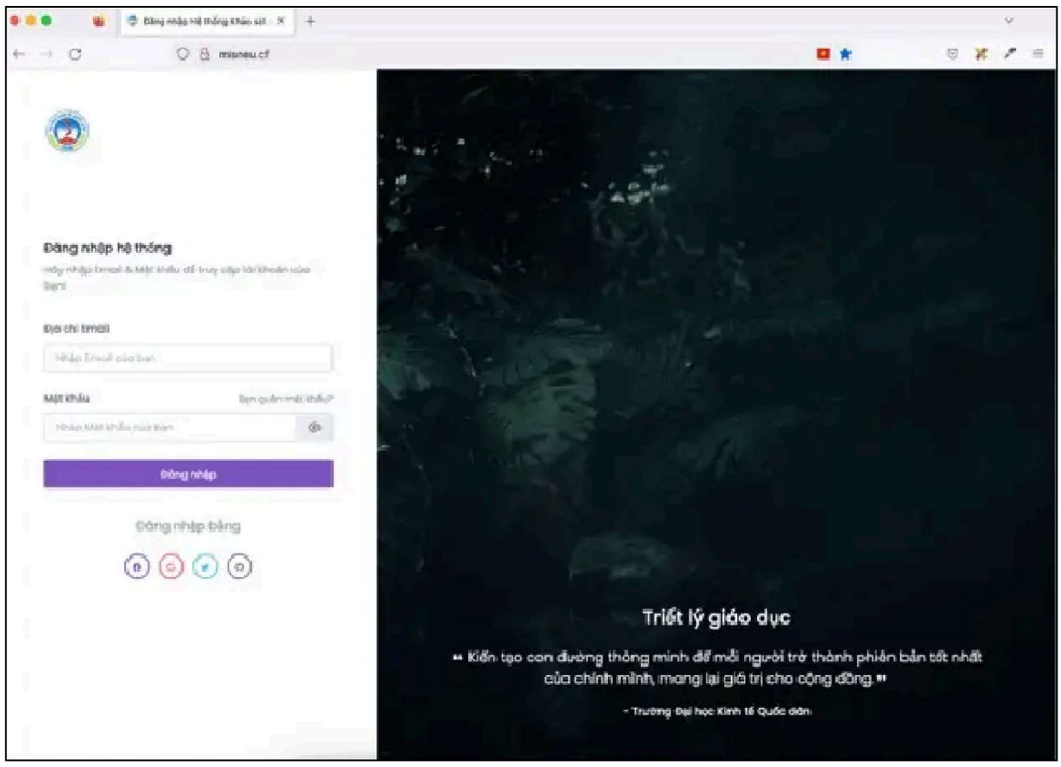
Bên cạnh đó, người dùng có thể sử dụng một số công cụ FTP Client chuyên nghiệp có tính phí *(như: CuteFTP,…)*. Hoặc miễn phí *(như: FTPRush, Cyberduck, Classic FTP, CoffeeCup Software, CoreFTP, …)*.

8.3 MỘT SỐ KỸ NĂNG CẦN THIẾT KHÁC KHI TRIỂN KHAI WEB

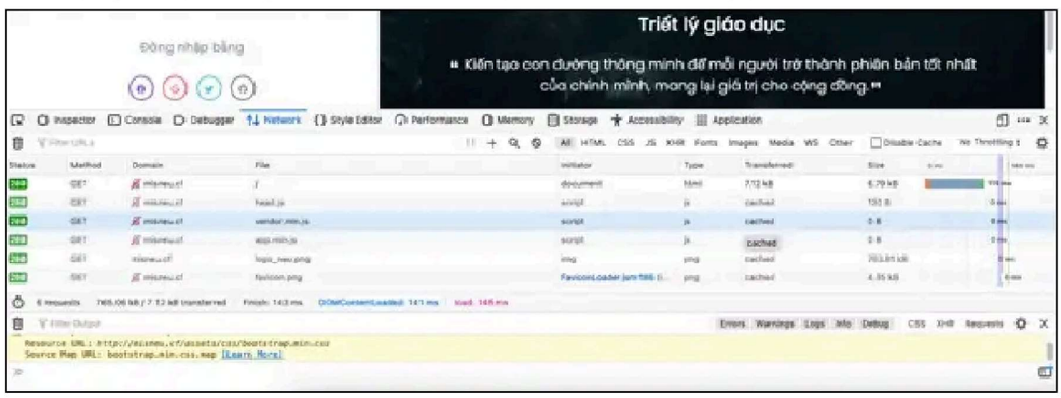
8.3.1 Kiểm thử Website sau khi tải lên Web host

Sau khi thực hiện (1) trỏ tên miền tới Web host và (2) chuyển mã nguồn trang Web từ máy tính cá nhân tới Web host, để kiểm tra Website đã hoạt động hay chưa, có thể tiến hành kiểm thử Website theo các bước sau:

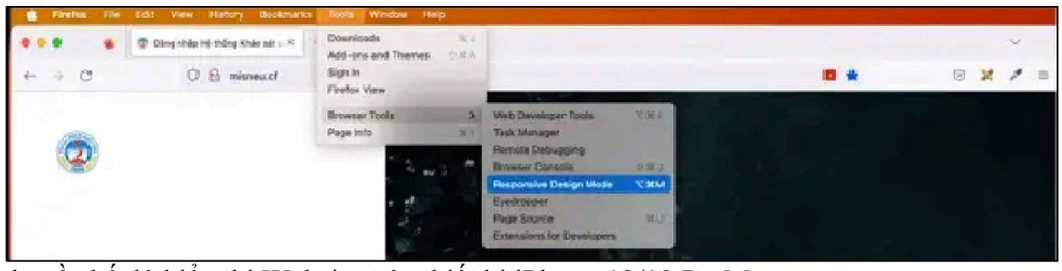
- Bước 1: Mở trình duyệt Web, nhập tên miền và tiến hành truy cập Website của cần kiểm thử.



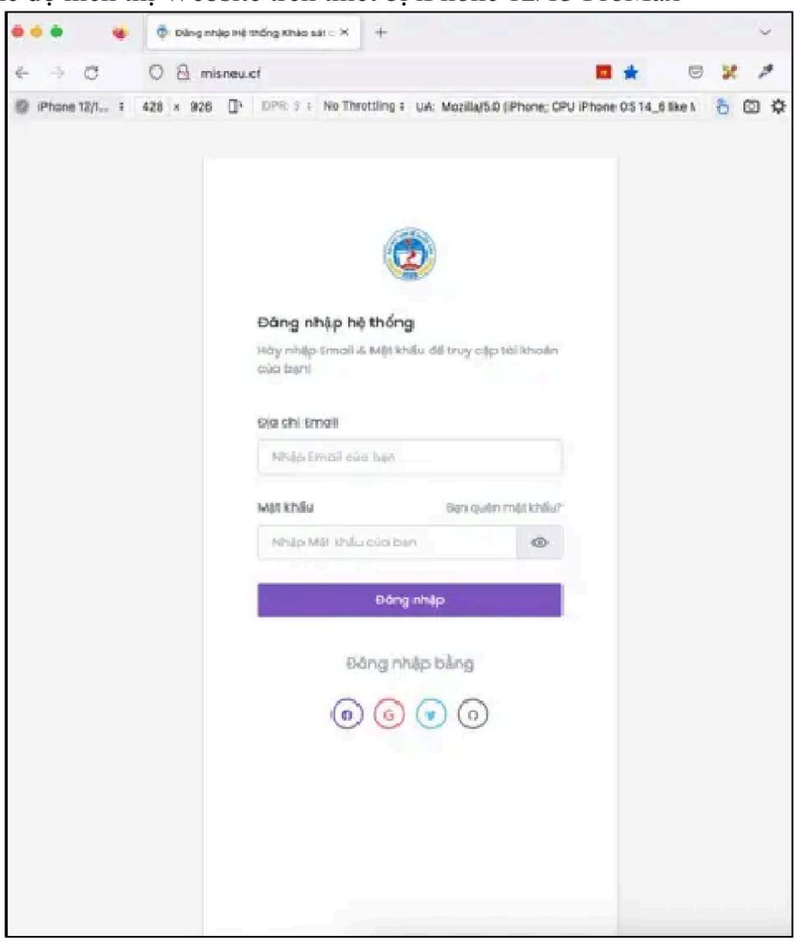
- Bước 2: Xem lại nội dung của trang và đảm bảo rằng giao diện hiển thị chính xác với mong muốn của người phát triển hệ thống.
  Ngoài ra, có thể kiểm tra tốc độ tải trang, rà soát xem việc tải các tài nguyên (file hình ảnh, CSS, JavaScript,…) có đầy đủ, chính xác và đảm bảo tốc độ cho phép hay không bằng cách ***click chuột phải → chọn “Inspect” → chọn Tab “Network” → click biểu tượng tải lại trang Web trên thanh địa chỉ của trình duyệt***.



- Bước 3: Kiểm tra tất cả các liên kết trên trang Web để đảm bảo các liên kết hoạt động chính xác.
- Bước 4: Thử nghiệm mở trang trong nhiều trình duyệt khác nhau để đảm bảo rằng trang Web hoạt động chính xác trong các trình duyệt đó.
  Bên cạnh đó, cũng có thể kiểm tra xem Website có hoạt động tốt trên thiết bị di động hay không thông qua lựa chọn ***Menu Tools → Browser Tools → tính năng Responsie Design Mode***.



Ví dụ về chế độ hiển thị Website trên thiết bị iPhone 12/13 ProMax

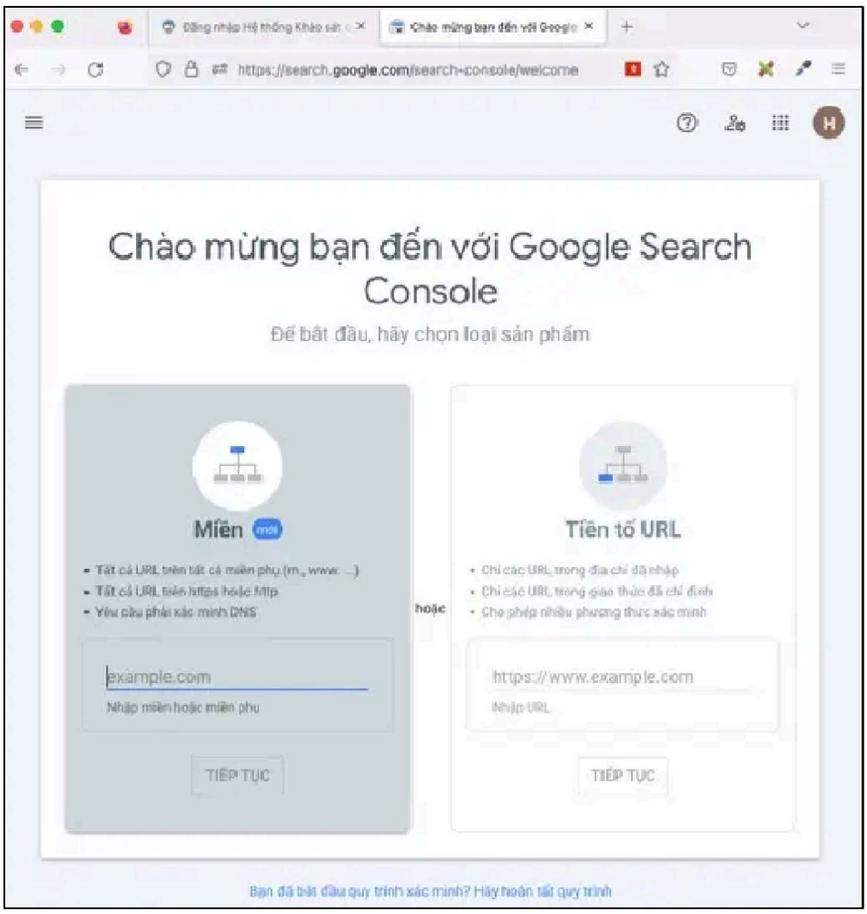


8.3.2 Khai báo Website tới các máy tìm kiếm

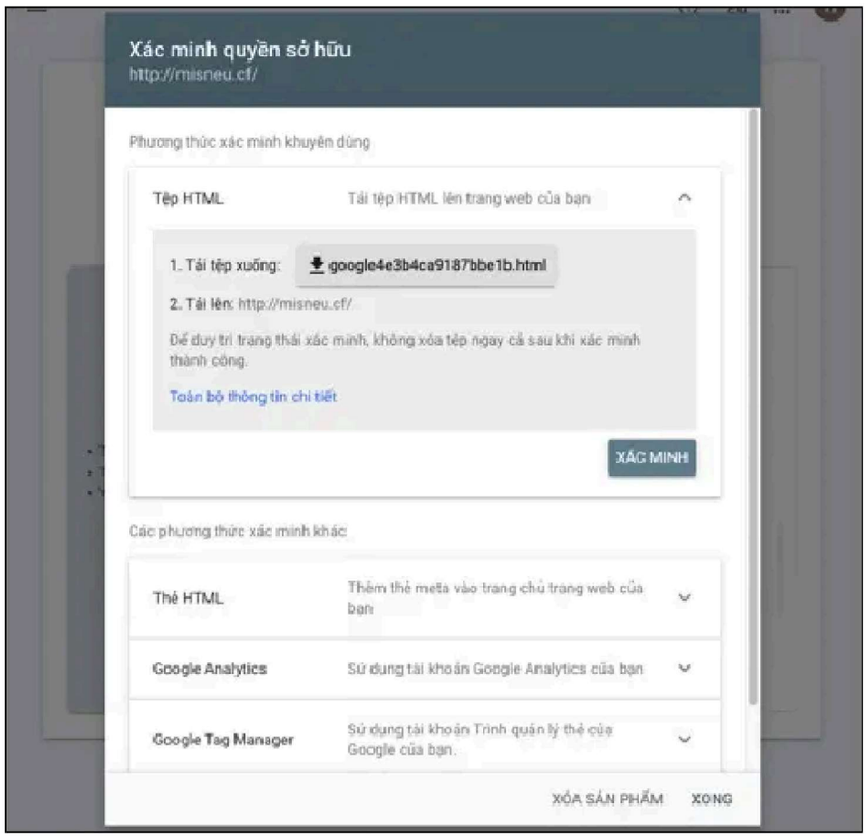
Sau khi phân tích, thiết kế, xây dựng, kiểm thử và đưa Website lên môi trường mạng Internet. Việc khai báo Website với các máy tìm kiếm chính (Google, Bing,…); sẽ giúp các máy tìm kiếm biết được sự tồn tại Website và đưa khách vãng lai truy cập Website. Để làm điều đó, có thể truy cập tới một số máy tìm kiếm nổi tiếng theo bảng sau:

Tên	Kiểu Website	Đường dẫn tới Website
Google	Máy tìm kiếm	https://www.google.com/webmasters/tools
Bing	Máy tìm kiếm	https://www.bing.com/webmasters

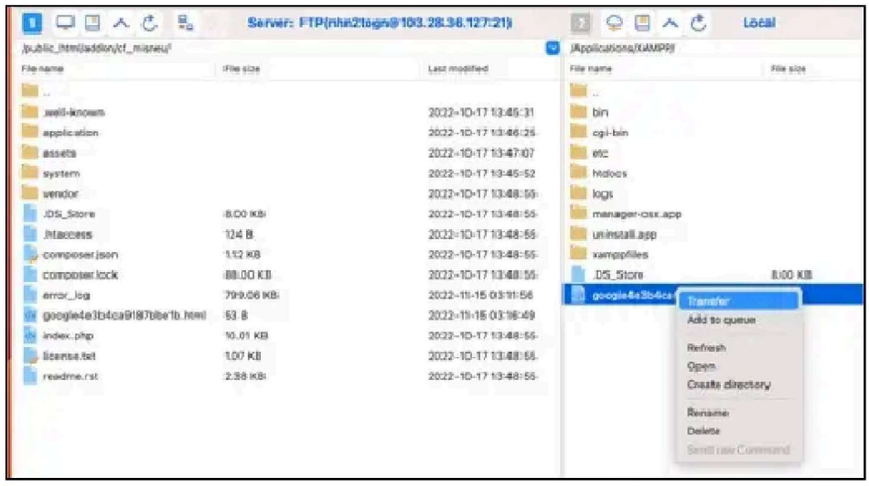
Các đường dẫn này sẽ chuyển tiếp tới trang phụ như trang Google phía dưới. Tại trang này, sẽ cần nhập địa chỉ Website muốn các Bot của máy tìm kiếm thu thập dữ liệu.



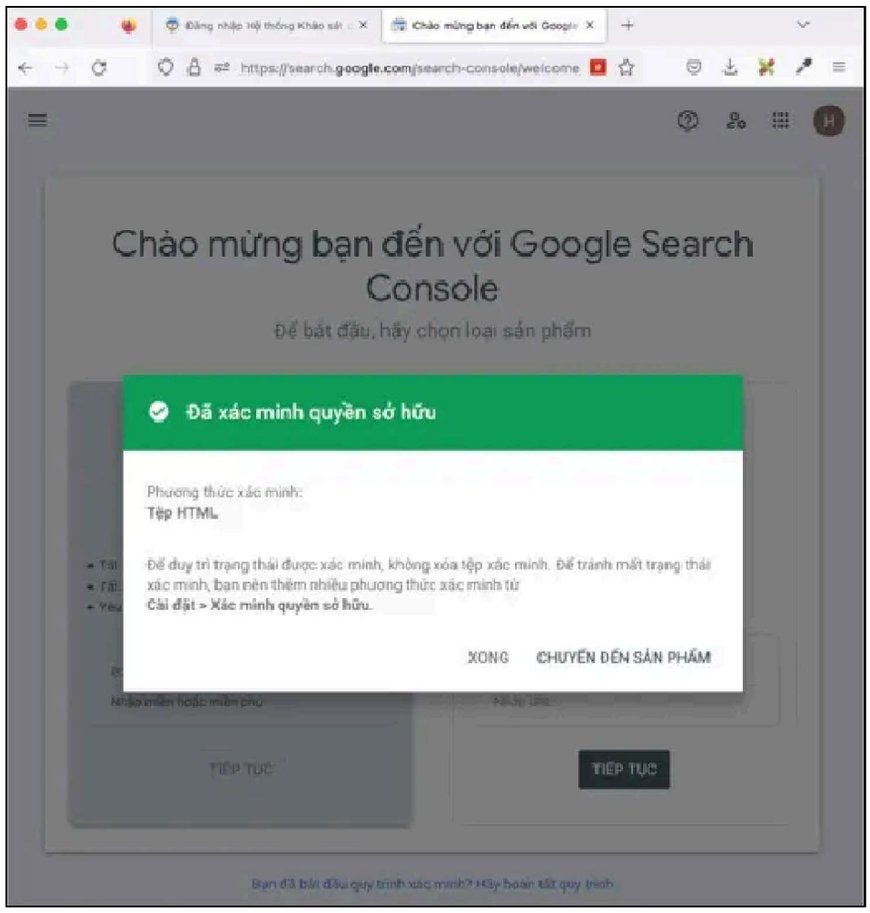
Tải file xác minh của máy tìm kiếm về máy tính cá nhân:



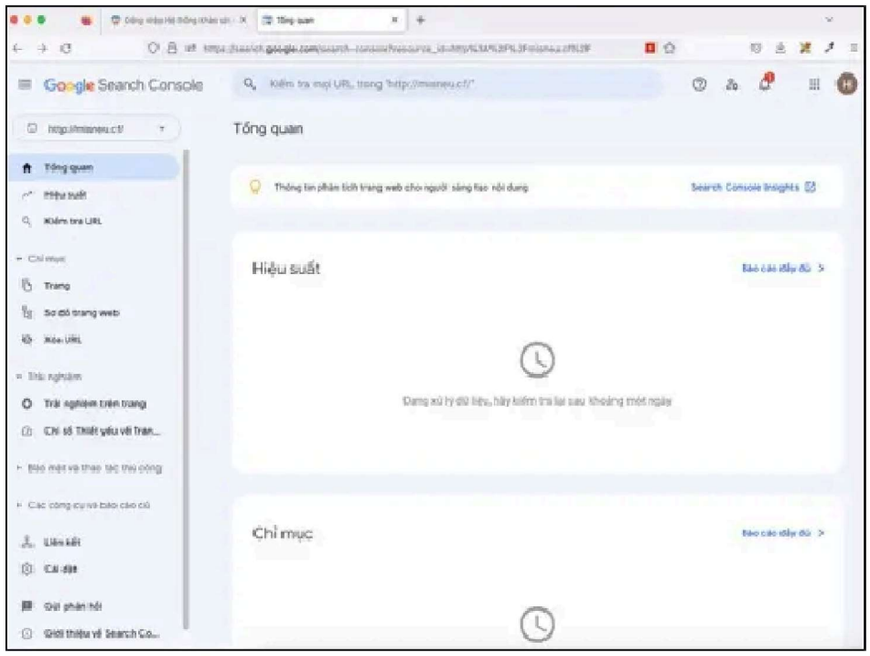
Chuyển file xác minh trên máy tính cá nhân lên thư mục gốc Website (*việc chuyển file có thể sử dụng ứng dụng FTP Client)*.



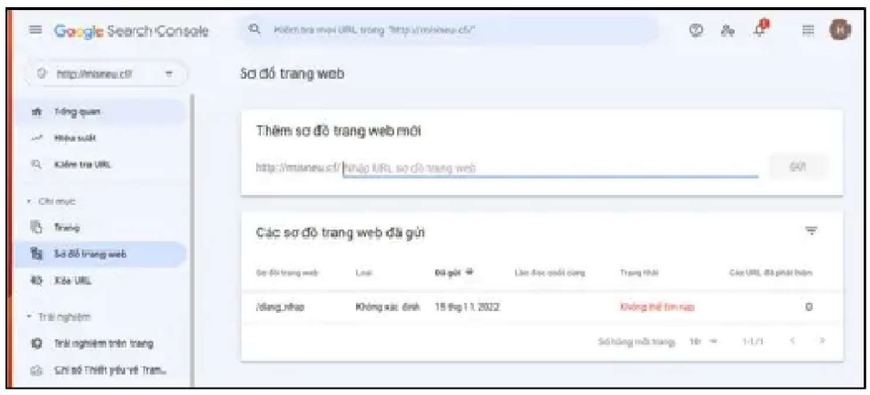
Sau khi chuyển file lên hoàn tất, quay lại Form và ấn vào nút "XÁC MINH". Nếu quá trình xác minh thành công, thông báo sau sẽ xuất hiện:



Người quản trị click vào **"Chuyển đến sản phẩm"**, sẽ được truy cập trang Google Search Console với đầy đủ công cụ để có thể thiết lập, theo rõi quá trình vận hành Website.



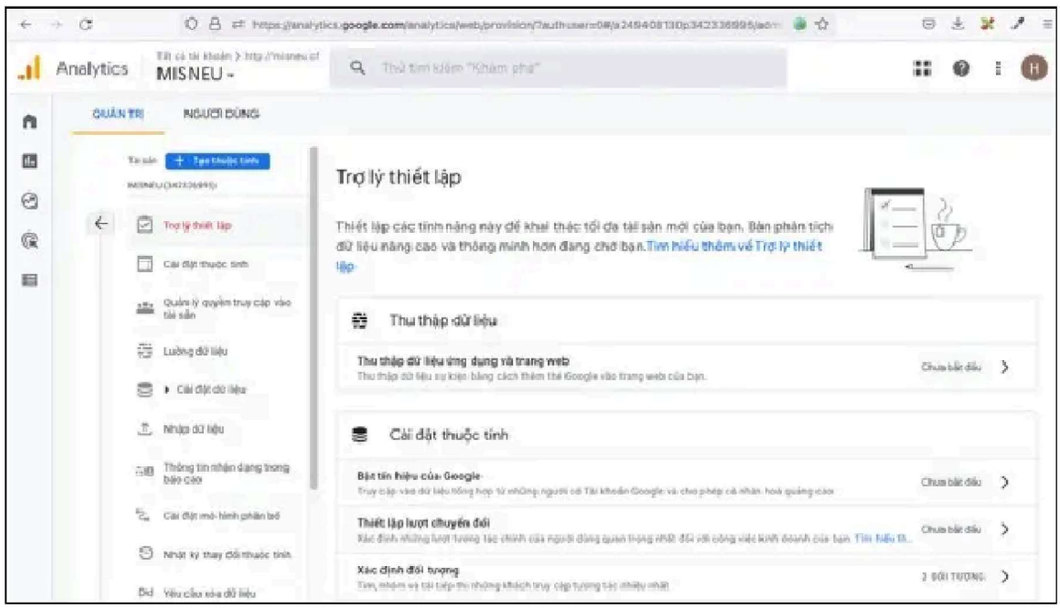
Với công cụ này, nếu Website được liên kết đúng cách với máy tìm kiếm, các Bot của máy tìm kiếm sẽ chủ động "thu thập thông tin" các trang còn lại trên Website bằng cách nhấp vào các liên kết trong trang Web.



Bên cạnh việc thu thập thông tin, các Bot của máy tìm kiếm sẽ chấm điểm trang Web. Những điểm số đó sẽ xác định trang Web xuất hiện như thế nào trong kết quả tìm kiếm và tất nhiên trang Web xuất hiện càng nhiều càng tốt. Vấn đề là các công cụ tìm kiếm sử dụng các thuật toán khác nhau để xác định điểm số của các trang Web. Thông thường, các công cụ tìm kiếm thay đổi thuật toán của họ theo thời gian mà không có bất kỳ thông báo trước. Ví dụ: một số công cụ tìm kiếm đánh giá tốt một trang Web nếu nó có liên kết đến nhiều trang Web khác; một số công cụ tìm kiếm thì đánh giá tốt một trang Web nếu nó được nhiều trang Web khác liên kết tới. Để tìm hiểu thêm các tiêu chí quan trọng trong thuật toán tính điểm được sử dụng, người quản trị có thể tham khảo thêm thông tin, gợi ý tối ưu hóa trang Web tại trang Web chính thức của các máy tìm kiếm. Ngoài ra, người quản trị cũng có thể tham khảo thêm các kỹ thuật tối ưu hóa công cụ tìm kiếm (SEO) được chia sẻ nhằm cập nhật cho trang Web.

Khi trang Web được khai báo trên các máy tìm kiếm, việc thực hiện cải tiến cho trang Web sẽ không cần phải thực hiện ngay vì các Bot của máy tìm kiếm định kỳ thu thập thông tin qua tất cả các trang và lập chỉ mục cho Website. Trong một số thời điểm, máy tìm kiếm sẽ chấm lại điểm của các trang.

Hầu hết các Web host đều cung cấp các gói thống kê cho biết khách truy cập đến từ đâu. Tuy vậy, cũng có thể sử dụng các giải pháp của bên thứ ba để làm điều đó như đăng ký tài khoản Google Analytics miễn phí (www.google.com.vn/analytics). Dữ liệu này có thể giúp tìm ra những trang đang hoạt động tốt và những trang không hoạt động. Vì vậy, có thể tìm ra những điểm cần thay đổi để cải thiện trang Web của mình.



8.3.3 Kiểm soát các trang được đánh chỉ mục và tìm kiếm

Đối với nhiều trang Web, có các trang và thậm chí các thư mục trong một trang có thể được tự động lập chỉ mục. Để kiểm soát được việc đó, người quản trị có thể tạo một tệp robots.txt đặt trong thư mục gốc của trang Web. Để giúp máy tìm kiếm hiểu lập chỉ mục trên Website như thế nào cho phù hợp, dưới đây là một số gợi ý.

```
#1: robots.txt - thông báo với các máy tìm kiếm không lập chỉ mục với các trang
trong thư mục cart.
    User-agent: *
    Disallow: /cart/

#2: robots.txt - thông báo với các máy tìm kiếm không lập chỉ mục các trang
trong 2 thư mục cart & private.
    User-agent: *
    Disallow: /cart/
    Disallow: /private/

#3: robots.txt - thông báo với các máy tìm kiếm không lập chỉ mục các trang
trong thư mục cart & 1 file cụ thể (private.html) trong thư mục backlist.
    User-agent: *
    Disallow: /cart/
    Disallow: /backlist/private.html
```

Bên cạnh đó, có thể sử dụng giải pháp khác là viết mã thẻ meta robot với nội dung được đặt thành "noindex" và “nofollow". Điều này có nghĩa là robot không nên lập chỉ mục trang và nó cũng không nên theo dõi bất kỳ liên kết nào trên trang khi nó thu thập thông tin liên quan tới Website.

```
<meta http-equiv="robots" content="noindex, nofollow">
```

Ngoài ra, một câu hỏi đặt ra là điều gì sẽ xảy ra nếu xóa một trang đã được lập chỉ mục trên Website? Kết quả là trang đó sẽ vẫn xuất hiện trong kết quả tìm kiếm cho đến khi Website được lập lại chỉ mục. Trong khi chờ đợi, nếu người dùng nhấp vào liên kết, trang sẽ không được tìm

thấy và hiển thị 404. Điều này dẫn tới vấn đề Website sẽ để lại thiện cảm không tốt, không tin cậy đối với người dùng. Giải pháp nhằm giải quyết vấn đề này là xóa nội dung khỏi trang, nhưng không xóa chính trang đó. Sau đó, có thể thêm một thẻ meta để chuyển hướng người dùng đến một trang khác.

```
<meta http-equiv="refresh" content="0" url=”../home.html”>
```

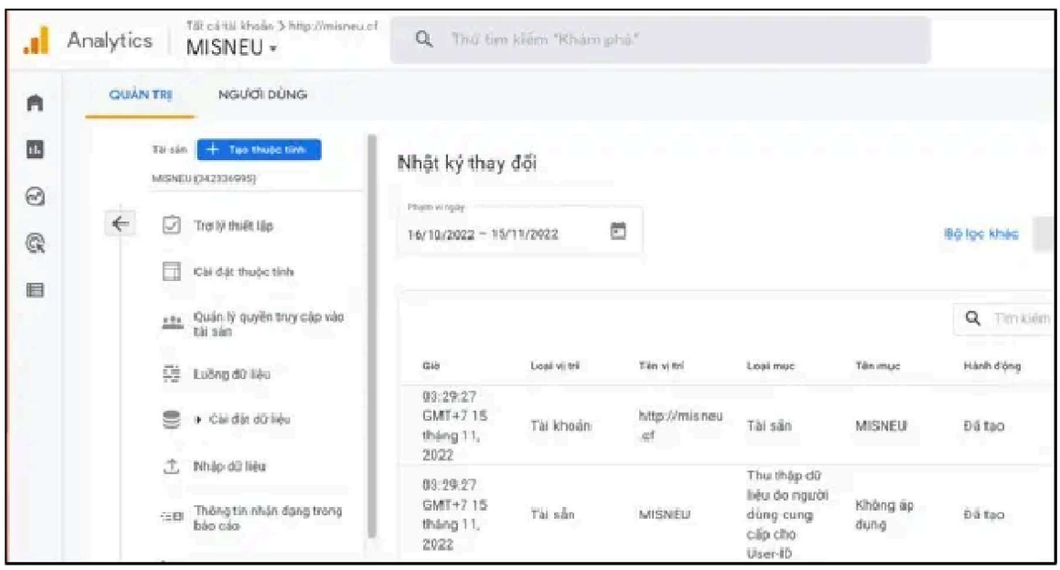
Trong thẻ này, nội dung (content) phải được đặt thành không (0) để chuyển hướng được thực hiện ngay lập tức (không có độ trễ). Sau đó, thuộc tính url sẽ chuyển người dùng tới trang mong muốn.

8.3.4 Bảo trì Website

Sau khi một Website được triển khai, được kiểm tra và sẵn sàng cho phép người dùng truy cập trên môi trường Internet thì vẫn cần thực hiện rà soát, cải thiện hiệu quả hoạt động của Website. Một số công cụ bên dưới giúp phân tích tổng thể trang Web, xác định các khía cạnh Website có thể cải thiện.

Công cụ	Địa chỉ URL
Google Webmaster Tools	https://www.google.com/webmasters/tools
Bing Webmaster Tools	https://www.bing.com/webmasters
SiteImprove	https://www.siteimprove.com
Hubspot's Marketing Grader	https://website.grader.com
FreeGrader	https://pagespeed.web.dev
Nibbler	https://nibbler.insites.com
Woorank	https://www.woorank.com

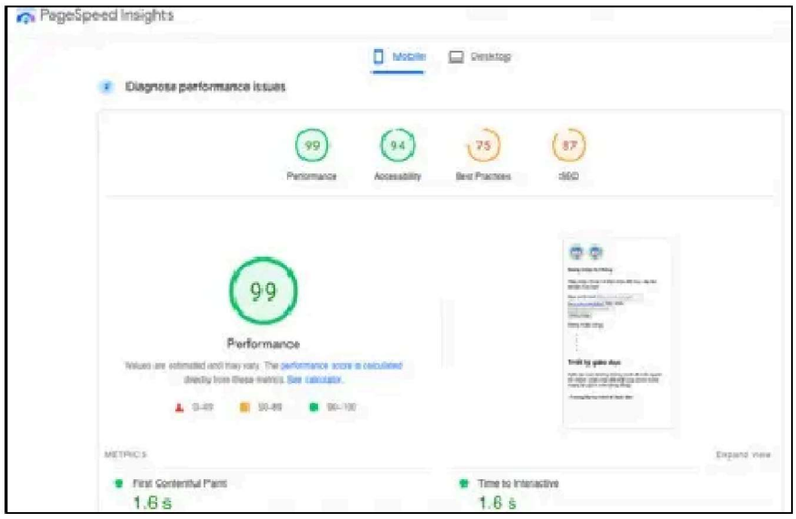
Công cụ phổ biến nhất để phân tích một trang Web là Google Webmaster. Công cụ này là một ứng dụng miễn phí nhằm theo dõi các lỗi Google thu thập thông tin, các vấn đề bảo mật, khả năng sử dụng trên thiết bị di động, các tham chiếu bị thiếu trong tệp robots.txt,... Nó cũng có thể đề xuất những cải tiến về HTML sẽ giúp tăng thứ hạng trên các công cụ tìm kiếm. Công cụ này đặc biệt hữu ích khi tích hợp nó với các công cụ khác của Google như Analytics và SiteSearch.

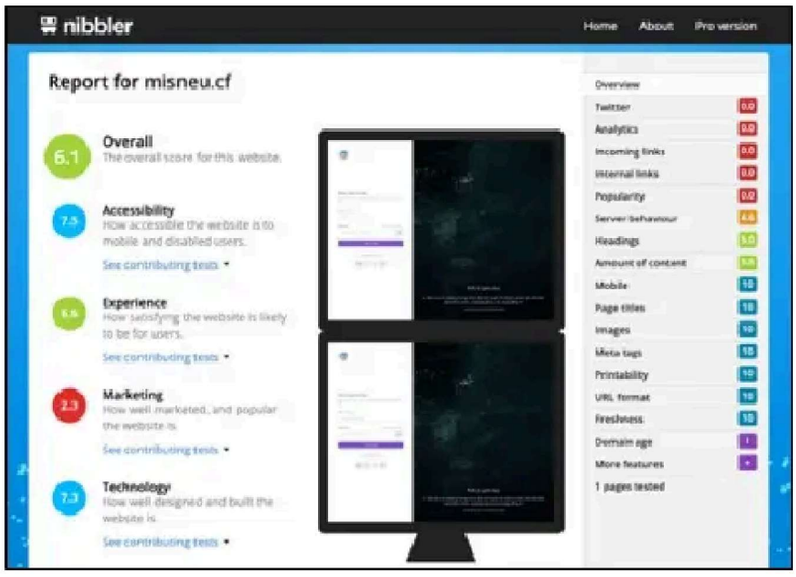


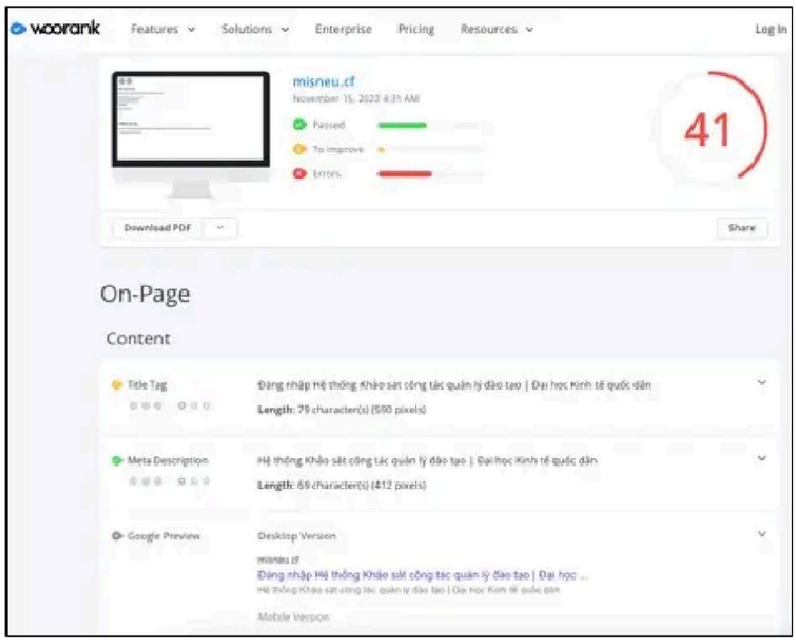
Nếu quản trị một trang Web "lớn", người quản trị có thể sử dụng một công cụ mạnh mẽ là SiteImprove. **SiteImprove** là một ứng dụng tính phí thực hiện thu thập dữ liệu chuyên sâu về trang Web như báo cáo lại các liên kết bị hỏng, các lỗi chính tả, các vấn đề về SEO và khả năng truy cập,... Trang quản trị của ứng dụng này rất dễ sử dụng và hỗ trợ khách hàng rất tốt. SiteImprove khá đắt nhưng rất xứng đáng cho tổ chức phụ thuộc nhiều vào kết quả phân tích trực tuyến của họ.

Hubspot's Marketing Grader đặc biệt hữu ích để kiểm tra hiệu quả tiếp thị trên mạng xã hội liên quan tới trang Web. Nó giám sát và báo cáo về sự hiện diện trên mạng xã hội của trang Web, bao gồm hoạt động tin tức, các chiến dịch tiếp thị qua email và thậm chí cả khả năng phản hồi trên thiết bị di động.

Bên cạnh đó, một số công cụ như FreeGrader, Nibbler và Woorank sẽ cung cấp các kiểm tra cơ bản về khả năng tiếp cận, SEO, trải nghiệm người dùng, khả năng phản hồi trên thiết bị di động và hơn thế nữa. Các công cụ này miễn phí, cung cấp ý tưởng về một số vấn đề nên và không nên thực hiện đối với Website được kiểm tra.


PageSpeed Insights
Mobile
Desktop
Diagnose performance issues
99
Performance
94
Accessibility
75
Best Practices
87
SEO
99
Performance
Values are estimated and may vary. The performance score is calculated directly from these metrics. See calculator.
METRICS
Expand view
First Contentful Paint
1.6 s
Time to Interactive
1.6 s

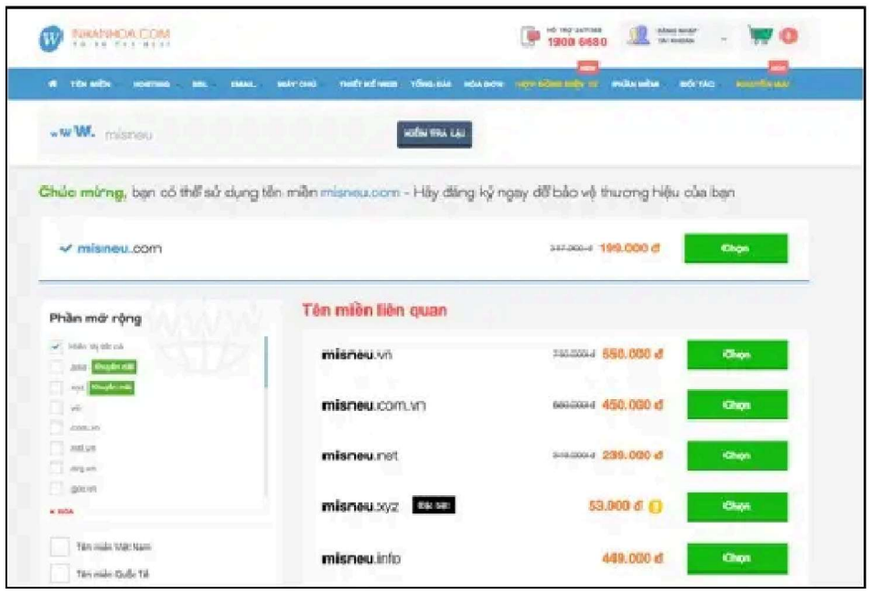

nibbler
Home
About
Pro version
Report for misneu.cf
Overall
The overall score for this website.
6.1
Accessibility
How accessible the website is to mobile and disabled users.
See contributing tests
Experience
How satisfying the website is likely to be for users.
See contributing tests
Marketing
How well marketed, and popular the website is.
See contributing tests
Technology
How well designed and built the website is.
See contributing tests
Overview
Twitter
Analytics
Incoming links
Internal links
Popularity
Server behaviour
Headings
Amount of content
Mobile
Page titles
Images
Meta tags
Printability
URL format
Freshness
Domain age
More features
1 pages tested



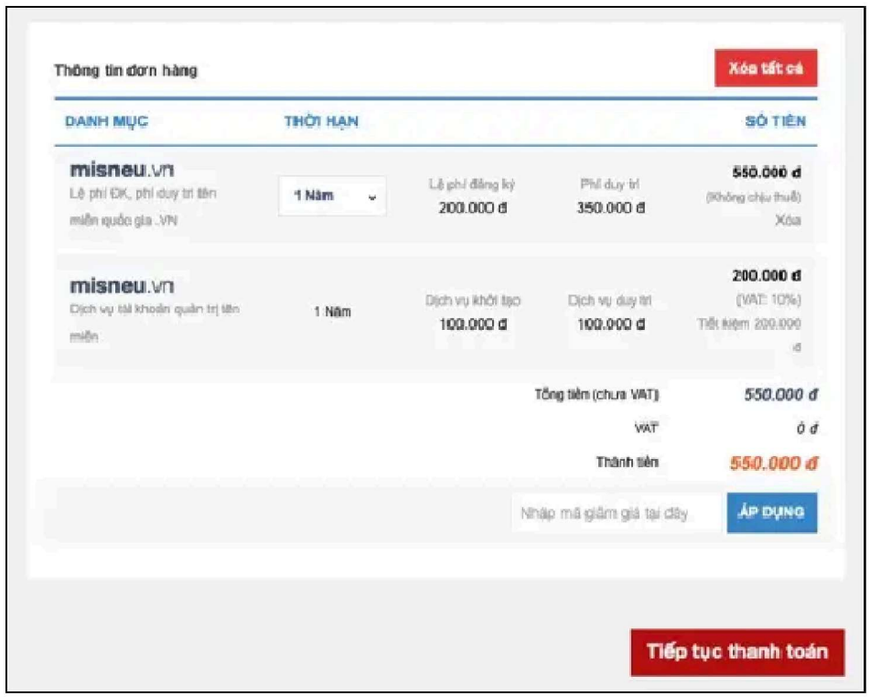
THỰC HÀNH

Bài 1. Mua tên miền

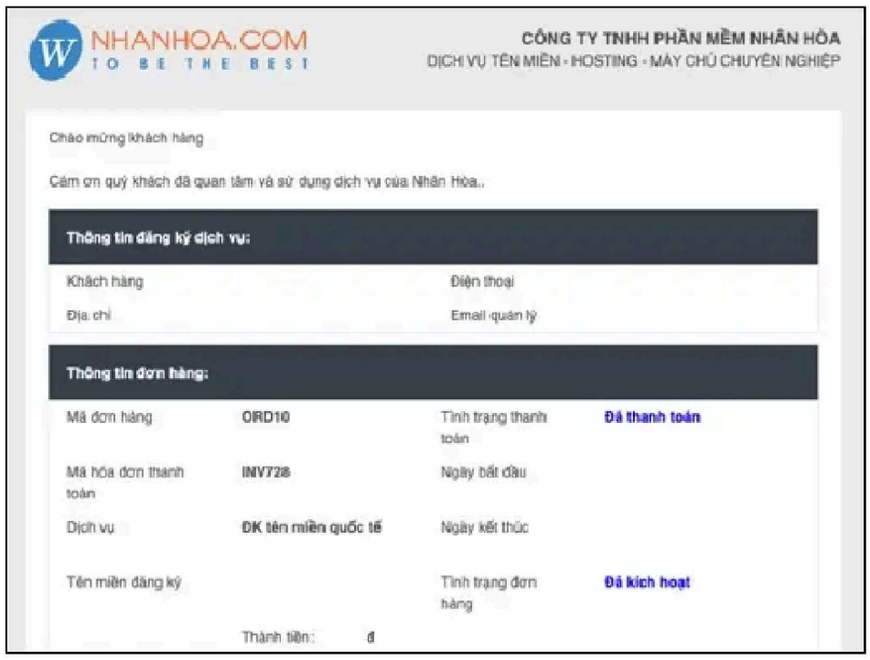
Bước 1: nhập tên miền muốn mua, kiểm tra xem tên miền dự định mua có được phép đăng ký hay không? Nếu được phép đăng ký, Click nút "Chọn".



Bước 2: Thanh toán chi phí mua tên miền, Click nút "Tiếp tục thanh toán". Tiến hành thanh toán để hoàn tất quá trình mua tên miền.



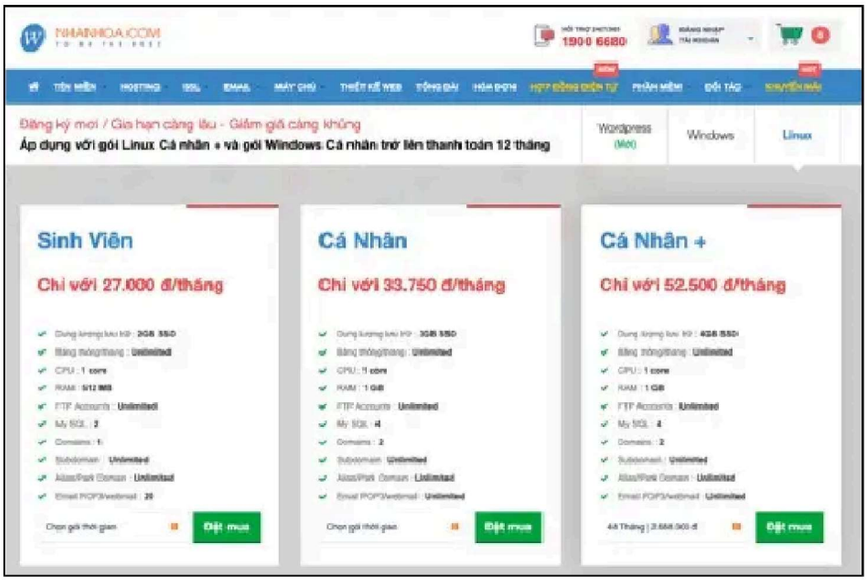
Bước 3: Nhận thư xác nhận đã mua tên miền thành công.



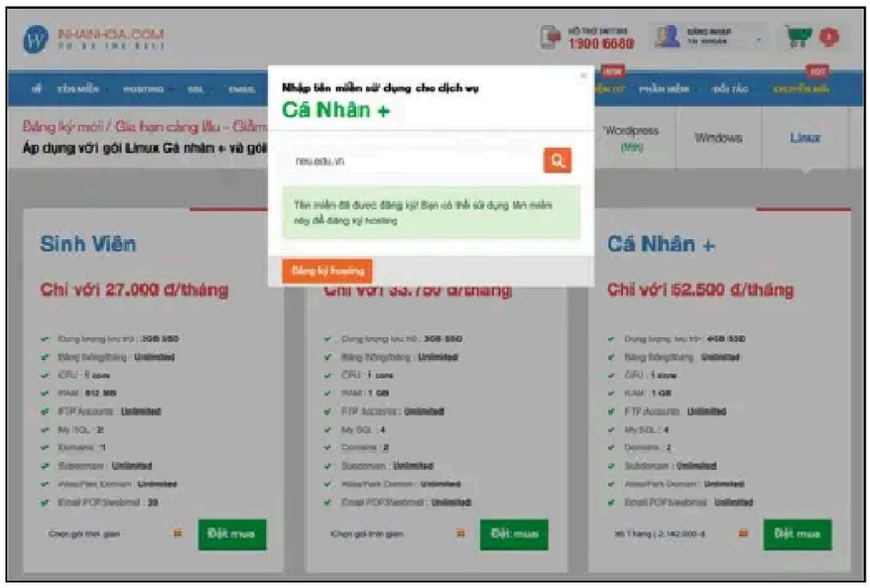
Bước 4: Trỏ tên miền tới nơi lưu trữ Website trên Web host.

Bài 2. Mua Web host

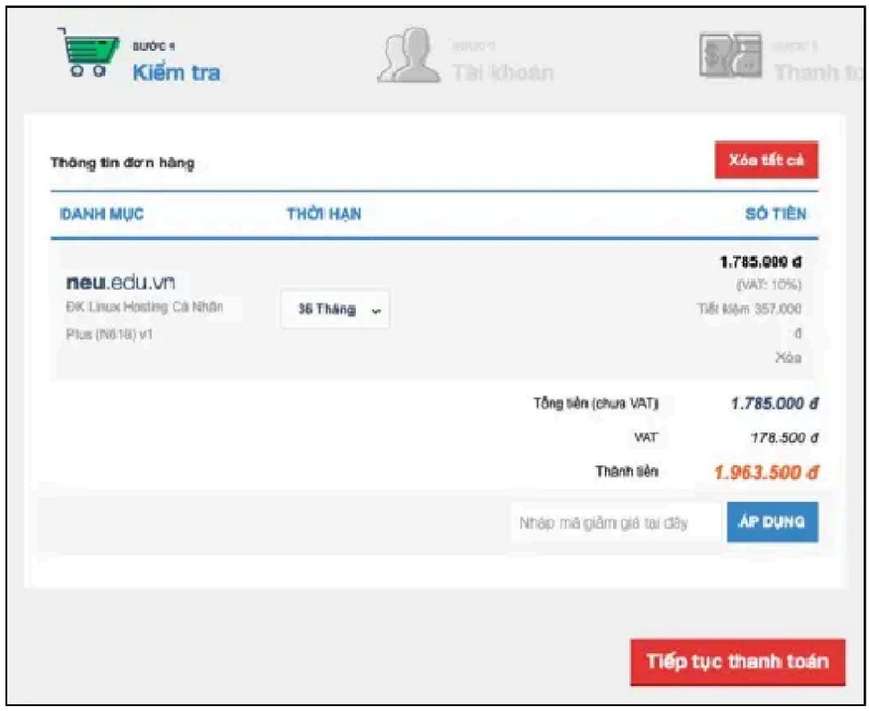
Bước 1: Lựa chọn gói Web host (gói, thời gian sử dụng gói) muốn mua.



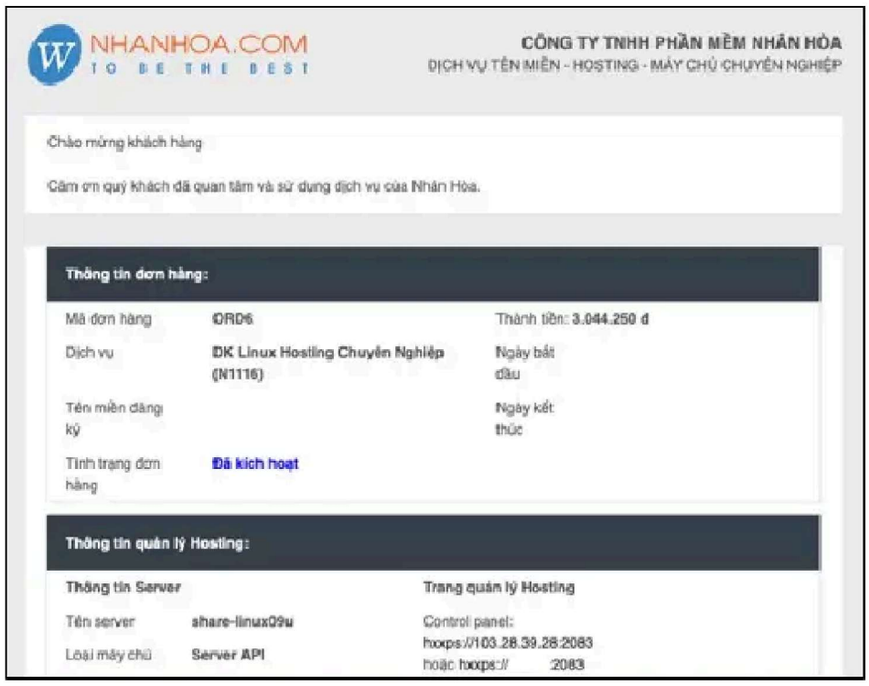
Bước 2: Click nút "Đặt mua" và nhập tên miền muốn được liên kết với Web host này. Sau đó, Click nút "Đăng ký hosting".



Bước 3: Thanh toán chi phí mua Web host; Click nút "Tiếp tục thanh toán". Tiến hành thanh toán để hoàn tất quá trình mua Web host.



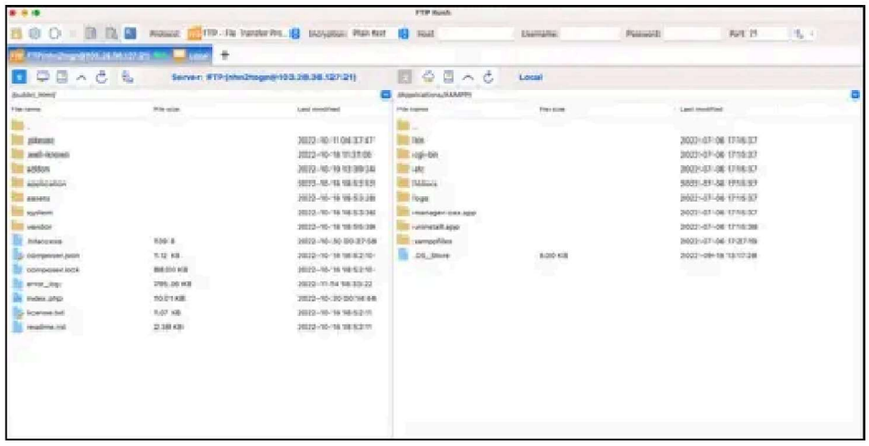
Bước 4: Nhận thư xác nhận đã mua tên miền thành công.



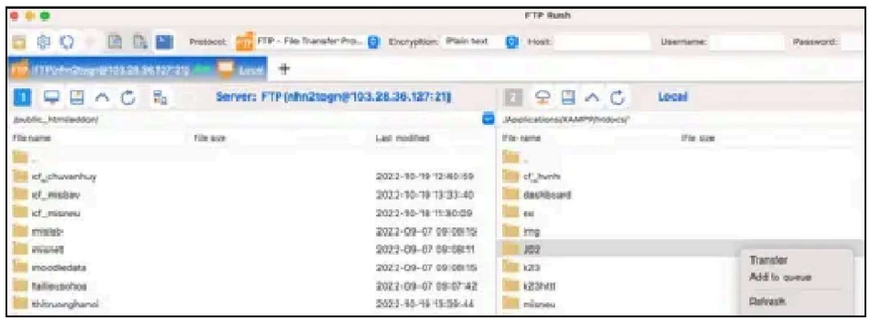
Bài 3. Chuyển mã nguồn trang Web lên Web host

Sau khi tải về và cài đặt ứng dụng FTP Client, có thể tiến hành chuyển toàn bộ mã nguồn trang Web từ máy tính cá nhân tới Web host thông qua các bước dưới đây.

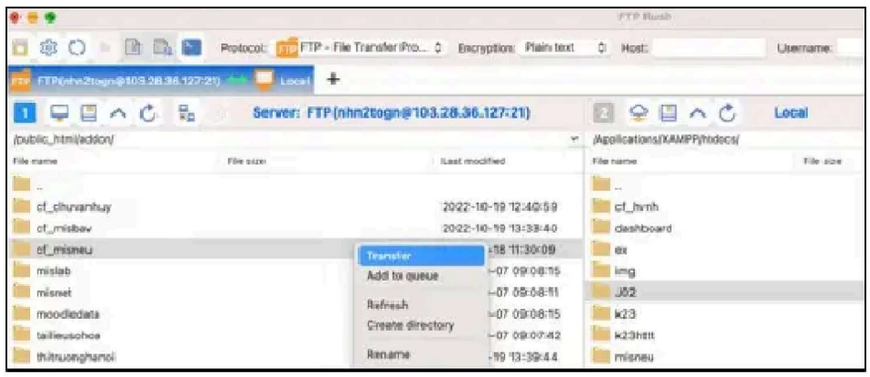
Bước 1: Nhập thông tin Web host (địa chỉ Web host, Username, Password) → nhấn nút "Connect" để có thể kết nối máy tính cá nhân với Web host.



Bước 2: Click chuột phải vào Thư mục (Folder) hoặc Tệp tin (File) trên máy tính cá nhân muốn chuyển sang Web host → chọn "Transfer".



Lưu ý: để sao lưu dữ liệu, có thể tải về mã nguồn mới nhất của Website trên Web host về máy tính cá nhân, Click chuột phải vào thư mục (hoặc tệp tin) trên Web host →nhấn nút "Transfer".



Bài 4. Triển khai Website lên Firebase

Vào firebase rồi lựa chọn mục Hosting để tiến hành xem các hướng dẫn deploy như dưới đây.

Sử dụng lệnh sau để cài firebase tools:

```
npm install -g firebase-tools
```

Đăng nhập vào filebase từ cmd:

```
firebase login
```

Khởi tạo thư mục:

```
firebase init
```

Tiến hành deploy:

```
firebase deploy
```

CÂU HỎI ÔN TẬP LÝ THUYẾT

1. **Tên miền Top-Level Domain (TLD) là gì?**

A. Phần trước dấu chấm của tên miền, thường là tên của tổ chức hoặc doanh nghiệp sở hữu

B. Phần sau dấu chấm của tên miền, chỉ định loại tên miền
C. Tên miền phổ biến như .com, .net, .org, .edu, v.v...
D. Cả A và C đều đúng

2. Tên miền phụ (subdomain) là gì?
A. Tên miền mà có chứa các ký tự đặc biệt như *, &, #, vv.
B. Tên miền được sử dụng để chia sẻ một website trên nhiều máy chủ
C. Tên miền được tạo ra bên trong tên miền chính để chỉ định một khu vực cụ thể của website
D. Tất cả các phương án trên đều sai

3. Để chọn một dịch vụ hosting phù hợp, cần lưu ý đến yếu tố nào?
A. Giá cả, tốc độ, bảo mật
B. Tên miền, giao diện, nội dung
C. Công nghệ, kinh nghiệm, thương hiệu
D. Tính năng, số lượng truy cập, chất lượng hỗ trợ

4. Web Host có thể được cài đặt trên hệ điều hành nào?
A. Windows
B. Linux
C. MacOS
D. Tất cả các phương án đều đúng

5. Loại hosting phổ biến thường được sử dụng?
A. Shared hosting
B. VPS hosting
C. Dedicated hosting
D. Tất cả các phương án đều đúng

6. Trong lĩnh vực phát triển web, SEO là cụm từ được viết tắt bởi?

A. Search Engine Optimization
B. Site Engine Optimization
C. Social Engine Optimization
D. Search Engine Operation

7. Sử dụng từ khóa quá nhiều trên một trang web có ảnh hưởng gì đến SEO?
A. Tốt cho SEO
B. Không ảnh hưởng đến SEO
C. Có thể làm giảm hiệu quả SEO
D. Làm tăng hiệu quả SEO

8. Tại sao phải thực hiện bảo trì website?
A. Để tăng cường tính năng và hiệu suất của website.
B. Để tạo ra một giao diện mới cho website.
C. Để tăng doanh số bán hàng trên website.
D. Để tạo ra nhiều trang web mới.

9. Trong quá trình triển khai website, phương thức nào thường được sử dụng để tải các tệp mã nguồn từ máy cá nhân lên Web host?
A. FTP
B. SMTP
C. HTTP
D. HTTPS

10. Để giảm thiểu thời gian tải website, có thể sử dụng công nghệ gì để phân phối nội dung trên các máy chủ khác nhau?
A. Load balancing
B. Server-side rendering
C. Caching
D. Content Delivery Network (CDN)

BÀI TẬP TỰ THỰC HÀNH

Hãy truy cập trang https://vn.000Webhost.com và thực hiện:

- Tạo tài khoản ở Website trên nhằm mục đích khởi tạo một **tên miền** và **Web host** miễn phí phục vụ thử nghiệm.
- Cài đặt công cụ FileZilla Client và tiến hành tải toàn bộ dữ liệu Website (đã xây dựng ở bài toán chương trước) lên Web host.
- Kiểm thử Website xem đã hoạt động ổn định hay chưa? Chia sẻ các lỗi Website mắc phải nếu có.

Sử dụng các công cụ FreeGrader, Nibbler và Woorank để đánh giá Website của bạn. Cần phải làm gì để cải thiện Website?

TÀI LIỆU THAM KHẢO

[1] Anne Boehm, Zak Ruvalcaba (2018) Murach's HTML5 and CSS3, Murack.

[2] Responsive Web Design with HTML5 and CSS: Build future-proof responsive Websites using the latest HTML5 and CSS techniques, 3rd, (2020), Ben Frain, Packt Publishing

[3] Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics (2018), Jennifer Robbins, O'Reilly Media

[4] Building Websites All-in-One For Dummies (2012), David Karlins, For Dummies

PHỤ LỤC

PHỤ LỤC 1: BẢNG CÁC PHẦN TỬ HTML

A

a, 78
abbr, 68
address, 69
article, 62
aside, 63
audio, 99

B

b, 70
base, 59
body, 53
br, 71
button, 91

C

canvas, 99
caption, 73
circle, 98
cite, 69
code, 71

D

datalist, 86
dd, 78
del, 70
details, 62
div, 59
dl, 78
dt, 78

E

ellipse, 99
em, 70
embed, 101

F

fieldset, 93
figcaption, 63
figure, 63
footer, 63
form, 80

H

h1, 65
h2, 65
h3, 65
h4, 65
h5, 65
h6, 65
head, 56
header, 62
hr, 64
html, 49

I

i, 70
iframe, 101
img, 97
input, 82
ins, 70

K

kbd, 72

L

label, 93
legend, 93
li, 75
link, 57

M

main, 60
map, 97
mark, 70
meta, 58

N

nav, 63
noscript, 230

O

ol, 76
optgroup, 88
option, 86
output, 94

P

p, 67
picture, 196
polygon, 99
pre, 71
process, 94

Q

q, 68

R

rect, 98
rt, 69
ruby, 69

S

s, 70
samp, 71

script, 230
section, 61
select, 87
small, 70
source, 99, 197
span, 60
strong, 70
style, 109
sub, 70
summary, 62
sup, 69, 70
svg, 98

T

table, 72
tbody, 74
td, 72
text, 99
textarea, 85
tfoot, 74
th, 72
thead, 74
title, 57
tr, 72
track, 100

U

u, 70
ul, 75

V

var, 72
video, 100

PHỤ LỤC 2: BẢNG TRA CỨU THUỘC TÍNH CSS

A

accent-color, 140
align, 156
align-content, 178, 184
align-items, 176
align-self, 181
all, 111

B

background, 127
background-attachment, 127
background-color, 125
background-image, 141
background-image, 125
background-position, 126, 141
border, 129, 138
border-bottom-style, 128
border-collapse, 138
border-color, 128
border-left-style, 129
border-radius, 129
border-right-style, 128
border-spacing, 138
border-style, 127
border-top-style, 128
border-width, 128

C

caption-side, 138
clear, 170
color, 129, 140
content, 136
counter-increment, 136
counter-rese, 136

D

direction, 131

E

empty-cells, 138

F

flex, 180
flex-basis, 180
flex-direction, 175
flex-flow, 176
flex-grow, 180
flex-shink, 180
flex-wrap, 176
float, 157, 168
font, 133
font-family, 132
font-size, 133
font-style, 132
font-variant, 133
font-weight, 133

G

grid-area, 187
grid-column, 186
grid-column-end, 185
grid-column-end, 186
grid-column-gap, 185
grid-column-start, 185, 186
grid-gap, 185
grid-row, 186
grid-row-gap, 185
grid-template, 183
grid-template-areas, 184
grid-template-columns, 183
grid-template-rows, 183

H

height, 138, 146

J

justify-content, 177, 184

L

letter-spacing, 131
line-height, 131
list-style, 136
list-style-image, 136
list-style-postition, 136
list-style-type, 135

M

margin, 149
margin-bottom, 149
margin-left, 149
margin-right, 149
margin-top, 149
marker-offset, 136
max-height, 147
max-width, 147

O

order, 180
overflow, 166

P

padding, 138, 139, 153
padding-bottom, 153
padding-left, 153
padding-right, 153
padding-top, 153
postion, 160

R

resize, 141

T

table-layout, 138
text-align, 129, 138, 157
text-decoration, 130
text-indent, 130
text-shadow, 131
text-transform, 130
transition, 141

V

vertical-align, 138

W

width, 138, 139, 146
word-spacing, 131

Z

z-index, 165

PHỤ LỤC 3: BẢNG DANH MỤC TỪ DÀNH RIÊNG TRONG JAVASCRIPT

B

break, 249, 253

C

catch, 254
class, 257, 259
const, 235
continue, 253

D

delete, 245, 256
do, 250

E

else, 247
extends, 258

F

false, 239
finally, 254
for, 251
function, 243

I

if, 247
in, 252
instanceof, 240

L

let, 234

N

new, 244
null, 240

O

of, 252

R

return, 243

S

set, 259
static, 258
super, 258
switch, 248

T

this, 256
throw, 254
true, 239
try, 254
typeof, 240

V

var, 233, 249

W

while, 249

TÀI LIỆU THAM KHẢO

[1] Anne Boehm, Zak Ruvalcaba (2018) Murach’s HTML5 and CSS3, Murack.

[2] Responsive Web Design with HTML5 and CSS: Build future-proof responsive Websites using the latest HTML5 and CSS techniques, 3rd, (2020), Ben Frain, Packt Publishing

[3] Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics (2018), Jennifer Robbins, O'Reilly Media

[4] The Book of CSS3, 2nd Edition: A Developer's Guide to the Future of Web Design (2014), Peter Gasston, No Starch Press

[5] Pro CSS3 Layout Techniques (2016), Sam Hampton-Smith, Apress

[6] UX / UI Wireframe Design Sketchbook: Mobile, Tablet and Desktop templates for responsive designs with project planning, (2020), UX/UI Designer Books, Independently published

[7] The Definitive Guide: Master the World's Most-Used Programming Language (2020), David Flanagan, O'Reilly Media

[8] Professional JavaScript® for Web Developers (2020), Matt Frisbie, Wrox

[9] JavaScript and jQuery: Interactive Front-End Web Development (2014), Jon Duckett, Wiley

[10] Building Websites All-in-One For Dummies (2012), David Karlins, For Dummies

[11] Jump Start Sass: Get Up to Speed With Sass in a Weekend (2016), Hugo Giraudel, SitePoint

[12] Build a Website Now: A Beginner's Guide to Web Development: HTML, CSS and Bootstrap (2019), Riwanto Megosinarso, Independently published